Hypergraph-Based Combinatorial Optimization of Matrix-Vector Multiplication

Preliminary Exam — 4/16/2008

Michael Wolf

Color Scheme of Text

- Original contribution
- Future work
  - Ongoing research
  - Proposed research

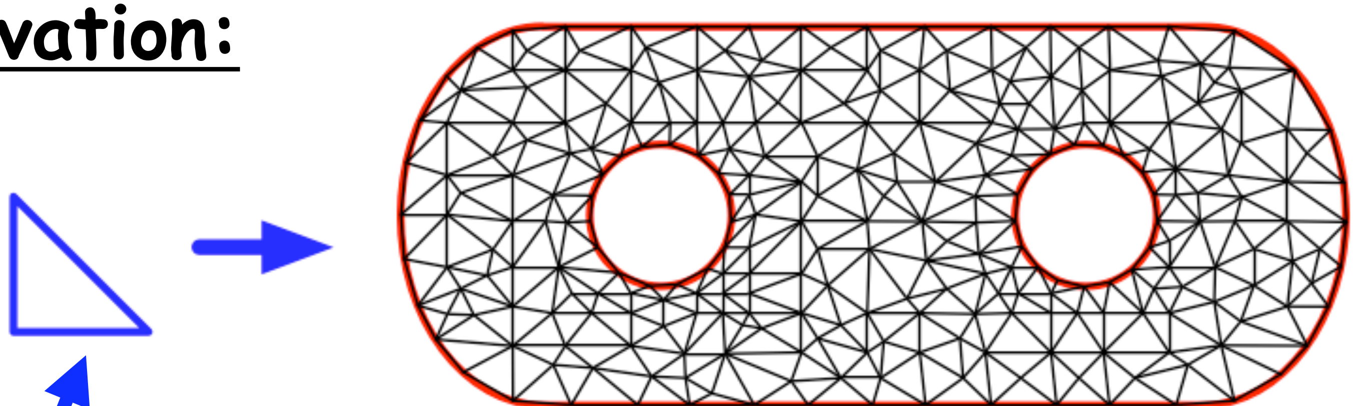
Combinatorial Optimization of Mat-Vec Multiplication

- Two main subtopics

- Serial matrix-vector multiplication
  - Reducing redundant operations
  - Dense relatively small matrices
- Parallel matrix-vector multiplication
  - Minimization of communication volume
  - Large, sparse matrices

Optimization of Serial Mat-Vec Multiplication

Motivation:



Based on reference element, generate code to optimize construction of local stiffness matrices

Can use optimized code for every element in domain

- Reducing redundant operations in building finite element (FE) stiffness matrices
  - Reuse optimized code when problem is rerun

Related Work

- Finite element "Compilers" (FEniCS project)
  - www.fenics.org
  - FIAT (automates generations of FEs)
  - FFC (variational forms -> code for evaluation)
- Following work by Kirby, et al., Texas Tech, University of Chicago on FErari
  - Optimization of FFC generated code to evaluate finite element matrices
  - Equivalent to optimizing matrix-vector product code

Matrix-Vector Multiplication

For 2D Laplace equation, we obtain following matrix-vector product to determine entries in local stiffness matrix

$$\mathbf{S}^e_{i,j} = y_{ni+j} = \mathbf{A}_{(ni+j,*)}\mathbf{x}$$

where

$$\mathbf{A}^T_{(ni+j,*)} = \begin{bmatrix} \left(\frac{\partial\phi_i}{\partial r}, \frac{\partial\phi_j}{\partial r}\right)_{\hat{e}} \\ \left(\frac{\partial\phi_i}{\partial r}, \frac{\partial\phi_j}{\partial s}\right)_{\hat{e}} \\ \left(\frac{\partial\phi_i}{\partial s}, \frac{\partial\phi_j}{\partial r}\right)_{\hat{e}} \\ \left(\frac{\partial\phi_i}{\partial s}, \frac{\partial\phi_j}{\partial s}\right)_{\hat{e}} \end{bmatrix}$$

Element independent

$$\mathbf{x} = \det(\mathbf{J}) \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} + \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} \\ \frac{\partial r}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial r}{\partial y}\frac{\partial s}{\partial y} \\ \frac{\partial s}{\partial x}\frac{\partial r}{\partial x} + \frac{\partial s}{\partial y}\frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial s}{\partial y}\frac{\partial s}{\partial y} \end{bmatrix}$$

Element dependent

Optimization Problem

Objective: Generate set of operations for computing matrix-vector product with minimal number of multiply-add pairs (MAPs)

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \mathbf{r_1}^T \\ \hline \mathbf{r_2}^T \\ \hline \vdots \\ \hline \mathbf{r_m}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{r_1}^T\mathbf{x} \\ \mathbf{r_2}^T\mathbf{x} \\ \vdots \\ \mathbf{r_m}^T\mathbf{x} \end{bmatrix}$$

Possible Optimizations - Collinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_2} = 1.5\mathbf{r_1}$$

Possible Optimizations - Colinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$\mathbf{r_2} = 1.5\mathbf{r_1} \Rightarrow y_2 = 1.5y_1$ 1 MAP

Possible Optimizations - Identical Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_3} = \mathbf{r_1} \Rightarrow y_3 = y_1$$ 0 MAPs

Special case when rows identical

Possible Optimizations - Partial Colinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_4} = 2.5\mathbf{r_1} + 8\mathbf{e_4}$$

Possible Optimizations - Partial Collinear Rows

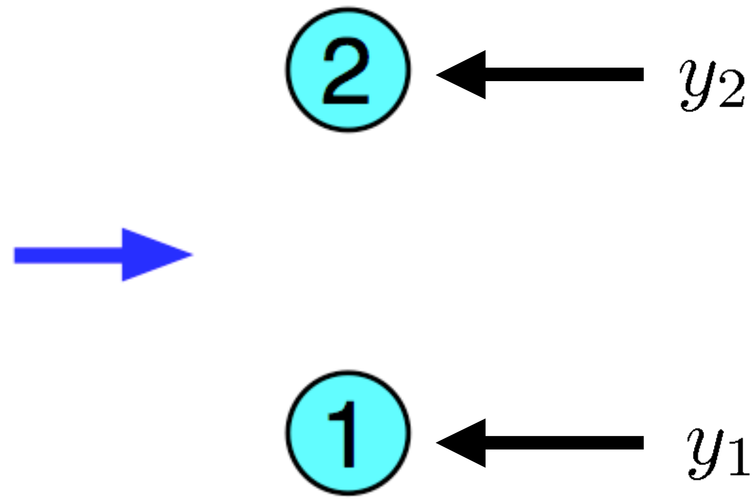
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_4} = 2.5\mathbf{r_1} + 8\mathbf{e_4} \Rightarrow y_4 = 2.5y_1 + 8x_4$$

2 MAPs

Graph Model - Resulting Vector Entry Vertices

$$\begin{array}{c|c:c:c:c|} \mathbf{r_1}^T & 8 & 4 & 4 & 8 \\ \hline \mathbf{r_2}^T & 0 & -2 & -2 & -4 \end{array}$$

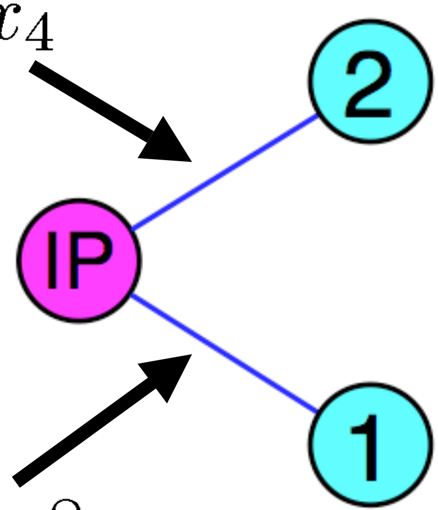


- Entries in resulting vector represented by vertices in graph model

Graph Model - Inner-Product Vertex and Edges

$$y_2 = -2x_2 - 2x_3 - 4x_4$$

$$\begin{array}{c} \mathbf{r_1}^T \\ \mathbf{r_2}^T \end{array} \begin{array}{|c:c:c:c|} \hline 8 & 4 & 4 & 8 \\ \hline 0 & -2 & -2 & -4 \\ \hline \end{array}$$

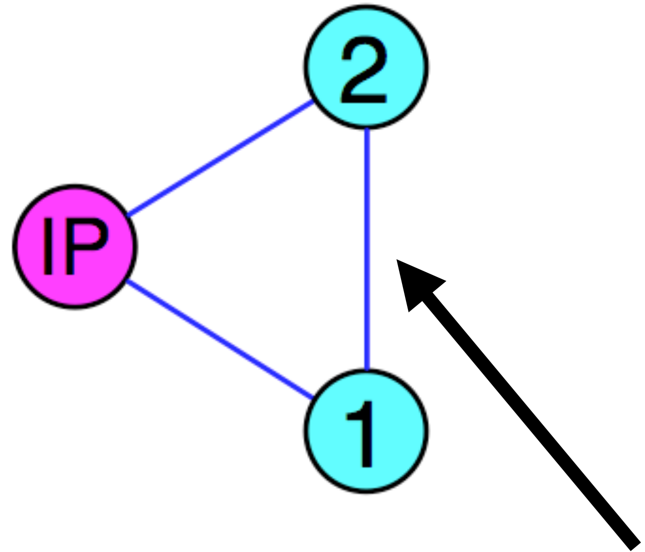


$$y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4$$

- Additional inner-product (IP) vertex
- Edges connect IP vertex to every other vertex, representing inner-product operation

Graph Model - Row Relationship Edges

$$\begin{array}{c} {\mathbf{r}_1}^T \\ {\mathbf{r}_2}^T \end{array} \begin{bmatrix} 8 & 4 & 4 & 8 \\ 0 & -2 & -2 & -4 \end{bmatrix}$$



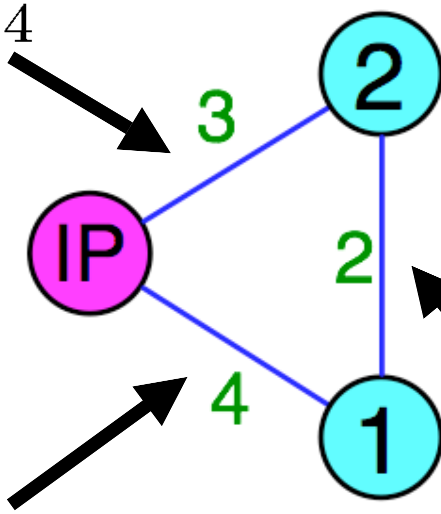
$$y_1 = -2y_2 + 8x_1$$
$$y_2 = -0.5y_1 + 4x_1$$

- Operations resulting from relationships between rows represented by edges between corresponding vertices

Graph Model - Edge Weights

$$y_2 = -2x_2 - 2x_3 - 4x_4$$

$\mathbf{r_1}^T$	8	4	4	8
$\mathbf{r_2}^T$	0	-2	-2	-4

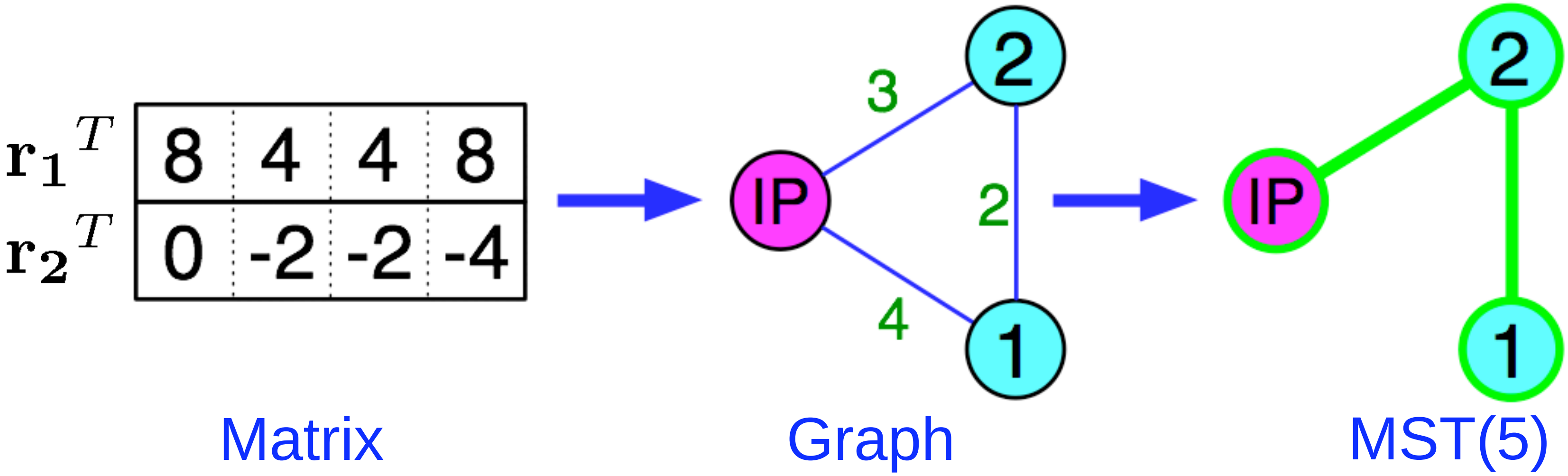


$$y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4$$

$$y_1 = -2y_2 + 8x_1$$
$$y_2 = -0.5y_1 + 4x_1$$

- Edge weights are MAP costs for operations

Graph Model Solution

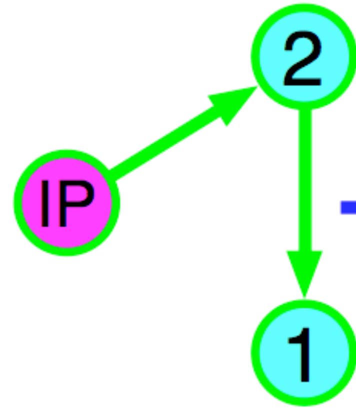


- Solution is minimum spanning tree (MST)
  - Minimum subgraph
  - Connected and spans vertices
  - Acyclic

Graph Model Solution

$$
\begin{array}{c} \mathbf{r_1}^T \\ \mathbf{r_2}^T \end{array}
\begin{bmatrix} 8 & 4 & 4 & 8 \\ 0 & -2 & -2 & -4 \end{bmatrix}
$$

Matrix



MST Traversal

$$
\boxed{\begin{aligned} y_2 &= -2x_2 - 2x_3 - 4x_4 \\ y_1 &= -2y_2 + 8x_1 \end{aligned}}
$$

Instructions

- Prim's algorithm to find MST (polynomial time)
- MST traversal yields operations to optimally compute (for these relationships) matrix-vector product

Graph Model Results - 2D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs
1	10	**7**
2	34	**14**
3	108	**43**
4	292	**152**
5	589	**366**
6	1070	**686**

60% decrease

- Graph model shows significant improvement over unoptimized algorithm

Graph Model Results - 2D Laplace Equation

Order	Unoptimized MAPs	FErari MAPs	Graph MAPs
1	10	**7**	**7**
2	34	15	**14**
3	108	45	**43**
4	292	176	**152**
5	589	443	**366**
6	1070	867	**686**

← 21% decrease

- Improved graph model shows significant improvement over FErari

Graph Model Results - 3D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs
1	21	**17**
2	177	**79**
3	789	**342**
4	2586	**1049**
5	7125	**3592**
6	16749	**8835**

59% decrease

- Again graph model requires significantly fewer MAPs than unoptimized algorithm

Graph Model Results - 3D Laplace Equation

Order	Unoptimized MAPs	FErari MAPs	Graph MAPs
1	21	–	**17**
2	177	101	**79**
3	789	370	**342**
4	2586	1118	**1049**
5	7125	–	**3592**
6	16749	–	**8835**

22% decrease

- Again graph model requires significantly fewer MAPs than FErari

Limitation of Graph Model

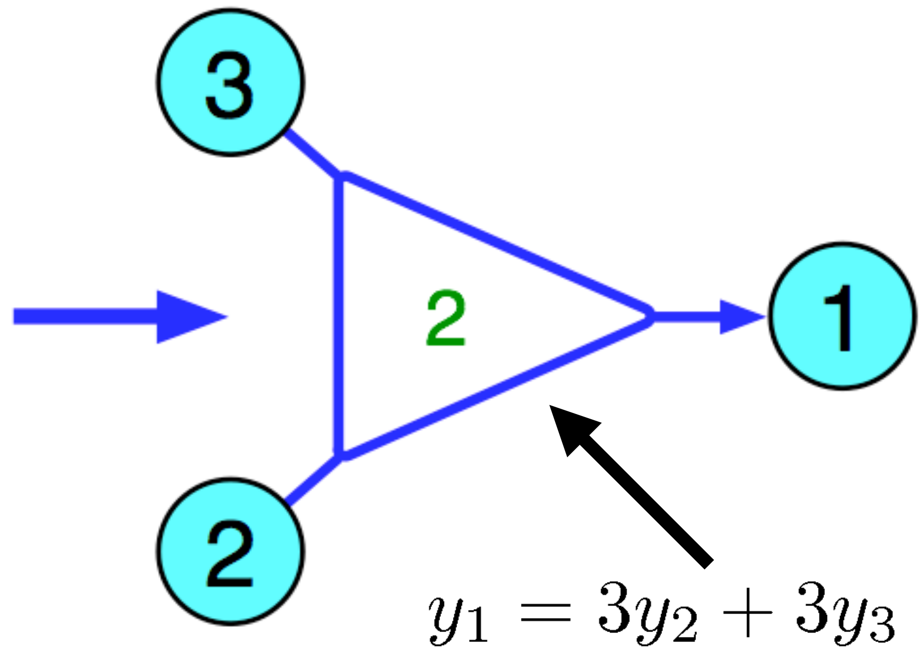
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 2 & 2 \\ 2 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_2} = 2\mathbf{r_3} + 2\mathbf{r_4} \Rightarrow y_2 = 2y_3 + 2y_4$$

- Edges connect 2 vertices
- Can represent only binary row relationships
- Cannot exploit linear dependency of more than two rows
- Thus, hypergraphs needed

Hypergraph Model

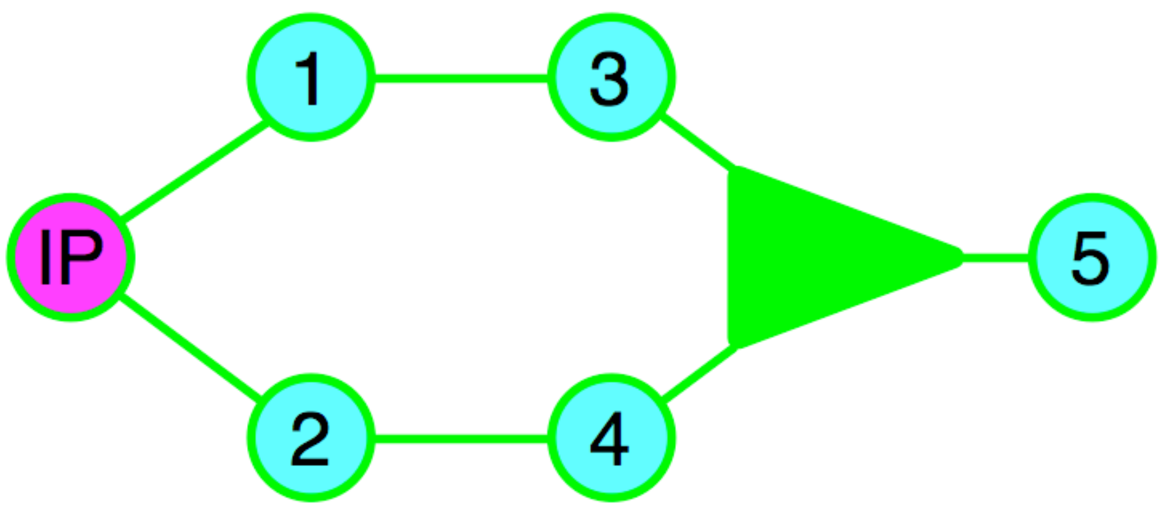
$$
\begin{array}{c|c:c:c:c|}
\mathbf{r_1}^T & 3 & 3 & 3 & 3 \\
\mathbf{r_2}^T & 1 & 1 & 0 & 0 \\
\mathbf{r_3}^T & 0 & 0 & 1 & 1
\end{array}
$$



- Same edges (2-vertex hyperedges) as graph model
- Additional higher cardinality hyperedges for more complicated relationships
  - Limiting to 3-vertex linear dependency hyperedges for this talk

Hypergraph Model

- Extended Prim's algorithm to include hyperedges
- Polynomial time algorithm
- Solution not necessarily a tree
  - {IP,1,3,5}
  - {IP,2,4,5}
- No guarantee of optimum solution



Hypergraph Model Results - 2D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs	HGraph MAPs
1	10	7	**6**
2	34	**14**	**14**
3	108	**43**	**43**
4	292	152	**150**
5	589	366	**363**
6	1070	**686**	**686**

- Hypergraph solution slightly better for some orders but not significantly better
- Graph algorithm solution close to optimal?
  - 3 Columns
  - Binary relationships may be good enough

Hypergraph Model Results - 3D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs	HGraph MAPs
1	21	**17**	**17**
2	177	79	**68**
3	789	342	**297**
4	2586	1049	**852**
5	7125	3592	**3261**
6	16749	8835	**8340**

19% additional decrease

- Hypergraph solution significantly better than graph solution for many orders

Future Work: New Hypergraph Method(s)

- Greedy modified Prim's algorithm yields suboptimal solutions for hypergraphs
- Want improved method that yields better (or optimal) solutions
  - Improved solution
  - Optimality of greedy solution
- First approach: integer programming method
  - Express valid hypergraph solution more formally
  - Exponential number of variables/constraints discouraging
- New approach: formulate as vertex ordering

Future Work: Vertex Ordering Method

- Order vertices
  - Roughly represents order of calculation for entries
- For given ordering, can determine optimal solution subhypergraph!
  - Greedy algorithm of selecting cheapest available hyperedge
  - Fast
- Ordering is challenging part
  - Traversal of greedy solution good starting point
  - Local refinement on starting point
- Develop global ordering method

Future Work: Hyperedge Detection/Construction

- Hyperedge detection/construction is bottleneck
- Currently brute force operation (nested loops)
  - e.g. $O(n^3)$ calls to coplanar detection kernel for n rows
- Detection kernel: originally SVD, now hybrid

Matrix	n	Orig Time (s)	Hybrid Time (s)
2DP5	231	9.1	1.4
2DP6	406	50.8	4.8
3DP3	210	4.8	0.4
3DP4	630	115.4	7.3
3DP5	1596	1921.9	117.5
3DP6	3570	26510.9	1248.2

- Improvement over brute force method
  - Better complexity than $O(n^3)$

Future Work: Hyperedge Pruning

- Hyperedge explosion
  - Over 10 million hyperedges for FE matrices
  - Hypergraphs too large to fit on one processor
- Most hyperedges won't be in optimal solution
- Want to prune as many as possible
- For example, currently prune
  - Hyperedge if weight greater or equal than number of nonzeros for all involved vertices
  - Coplanar (3 V) hyperedge if two of rows are collinear
- Need additional pruning heuristics
  - One possibility: use graph solution

Future Work: Miscellaneous

- Runtime of resulting operations
  - Preliminary studies show slight improvement
  - Not as good as MAP improvement
  - More complete study necessary
- Better instruction ordering
  - Currently do naive traversal of solution subgraph
  - Can do something more clever/cache-friendly
  - Solution is dependency graph

Sparse Matrix Partitioning

- Work with Dr. Erik Boman (SNL)
  - CSCAPES Institute
- Researched and developed two new two-dimensional methods
- If successful, will be implemented as part of new matrix partitioning suite in Zoltan

Parallel Sparse Matrix-Vector Multiplication

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} 1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 \\ 0 & 8 & 1 & 7 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 0 & 0 & 0 & 7 \\ 0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 \\ 4 & 0 & 0 & 0 & 3 & 1 & 3 & 0 \\ 0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 3 \\ 1 \\ 4 \\ 2 \\ 1 \end{bmatrix}$$

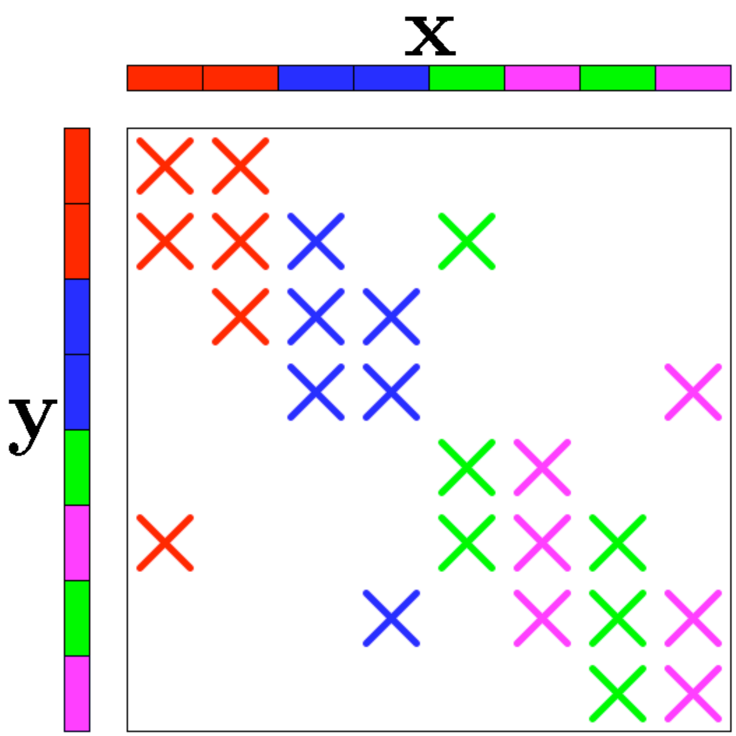
$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

- Partition matrix nonzeros
- Partition vectors

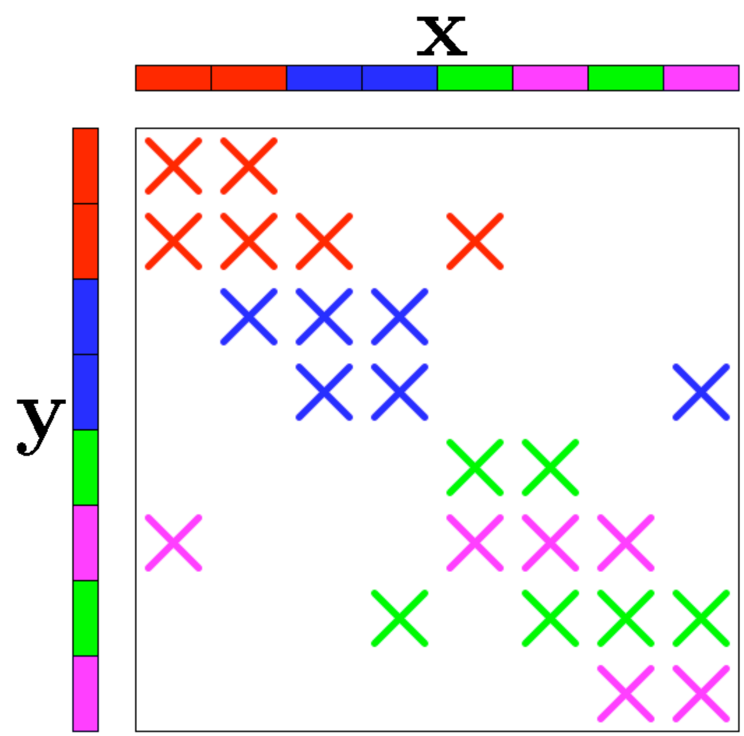
Objective

- Ideally we minimize total run-time
- Settle for easier objective
  - Work balanced
  - Minimize total communication volume
- Can partition matrices in different ways
  - 1-D
  - 2-D
- Can model problem in different ways
  - Graph
  - Bipartite graph
  - Hypergraph

1-D Partitioning



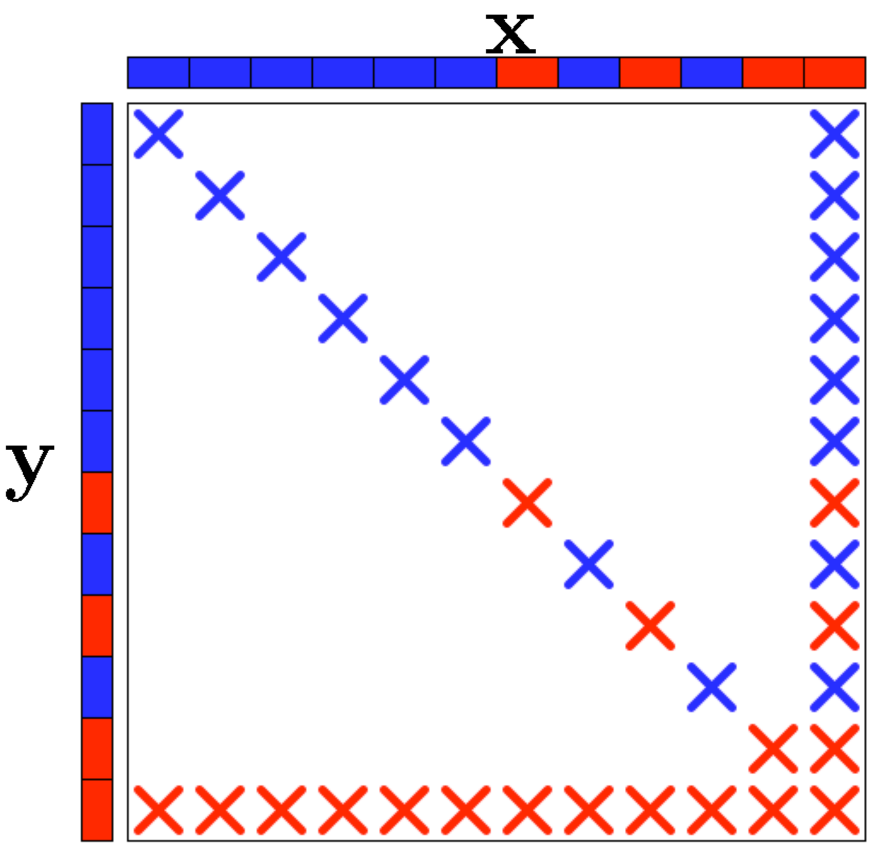
1-D Column



1-D Row

- Each process assigned nonzeros for set of columns
- Each process assigned nonzeros for set of rows

When 1-D Partitioning is Inadequate



"Arrowhead" matrix

n=12

nnz=34 (18,16)

volume = 9

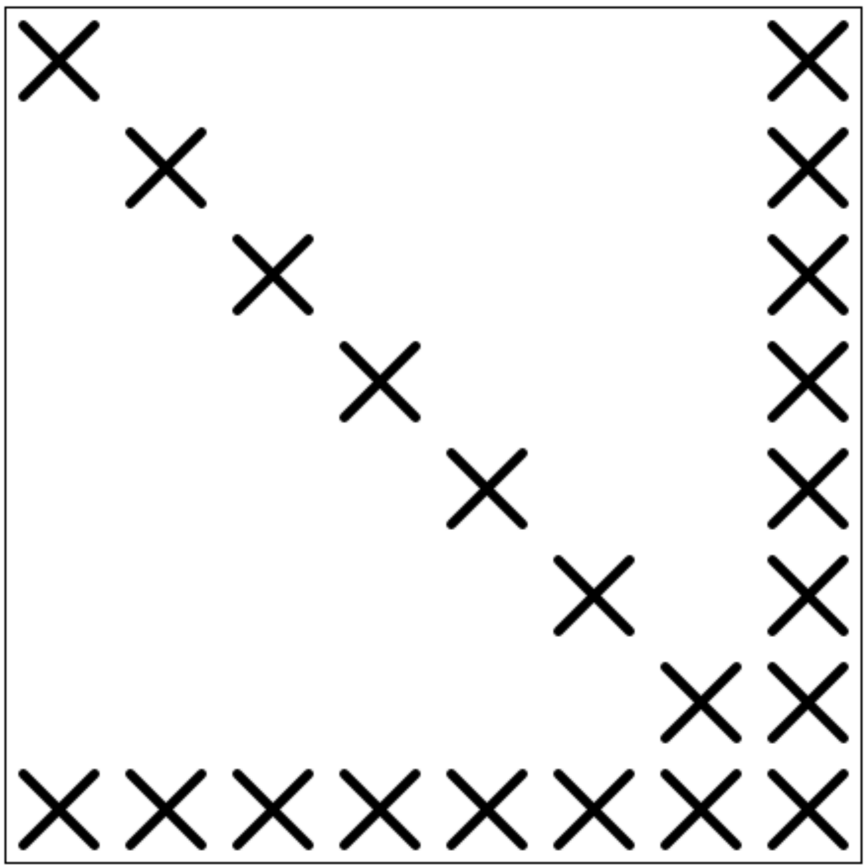
- For any 1-D bisection of nxn arrowhead matrix:
  - nnz = 3n-2
  - Volume ≈ (3/4)n
- O(k) volume partitioning possible

2-D Partitioning

- More flexibility in partitioning
- No particular part for given row or column
- More general sets of nonzeros assigned parts

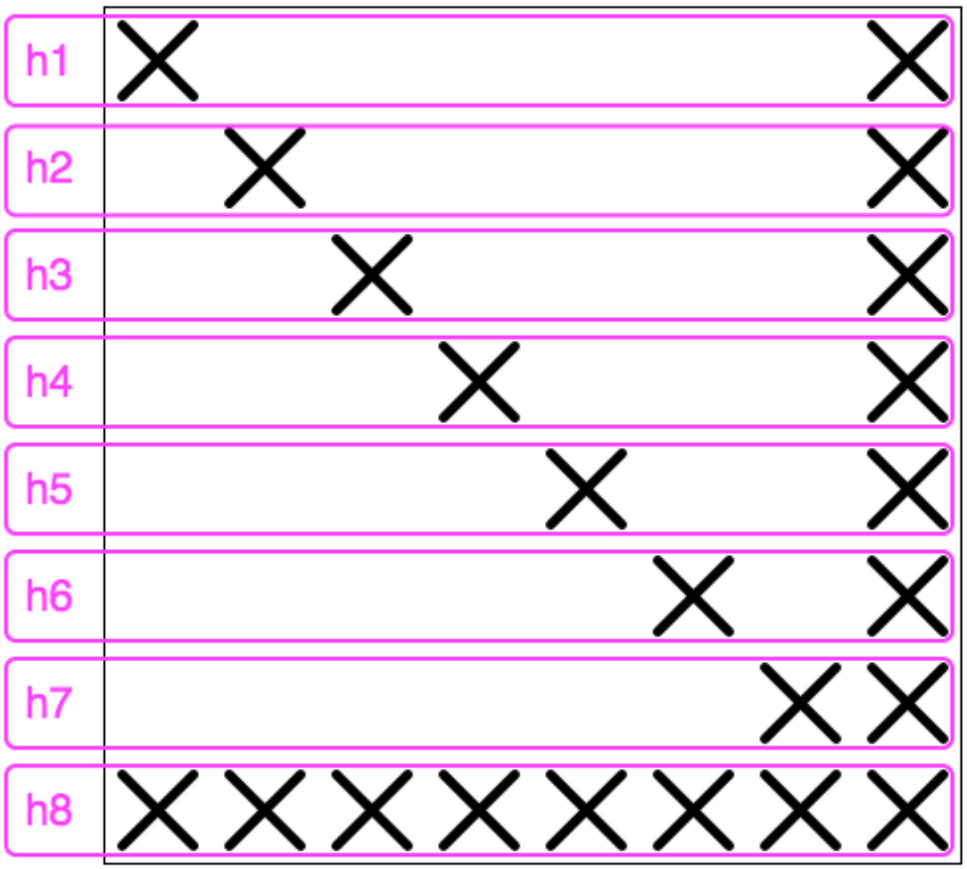
- Fine-grain hypergraph model
  - Ultimate flexibility
  - Assign each nz separately
- Corner symmetric partitioning method
- Graph model for symmetric 2-D partitioning
- Nested dissection symmetric partitioning method

Fine-Grain Hypergraph Model



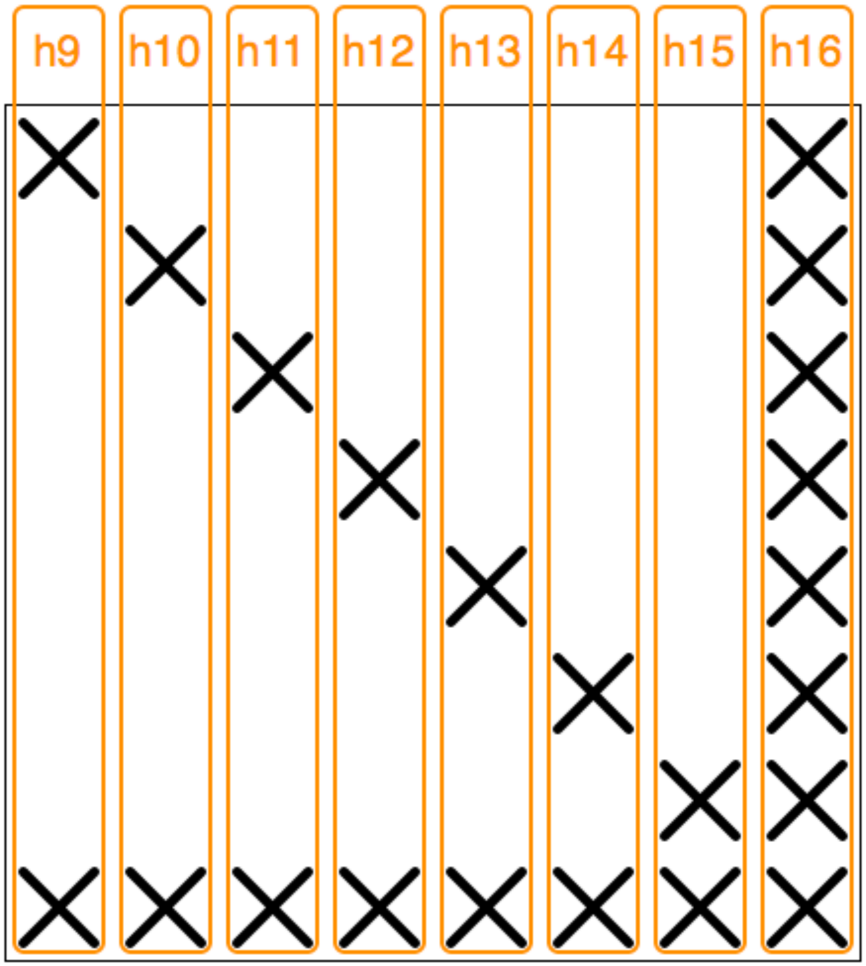
- Catalyurek and Aykanat (2001)
- Nonzeros represented by vertices in hypergraph

Fine-Grain Hypergraph Model



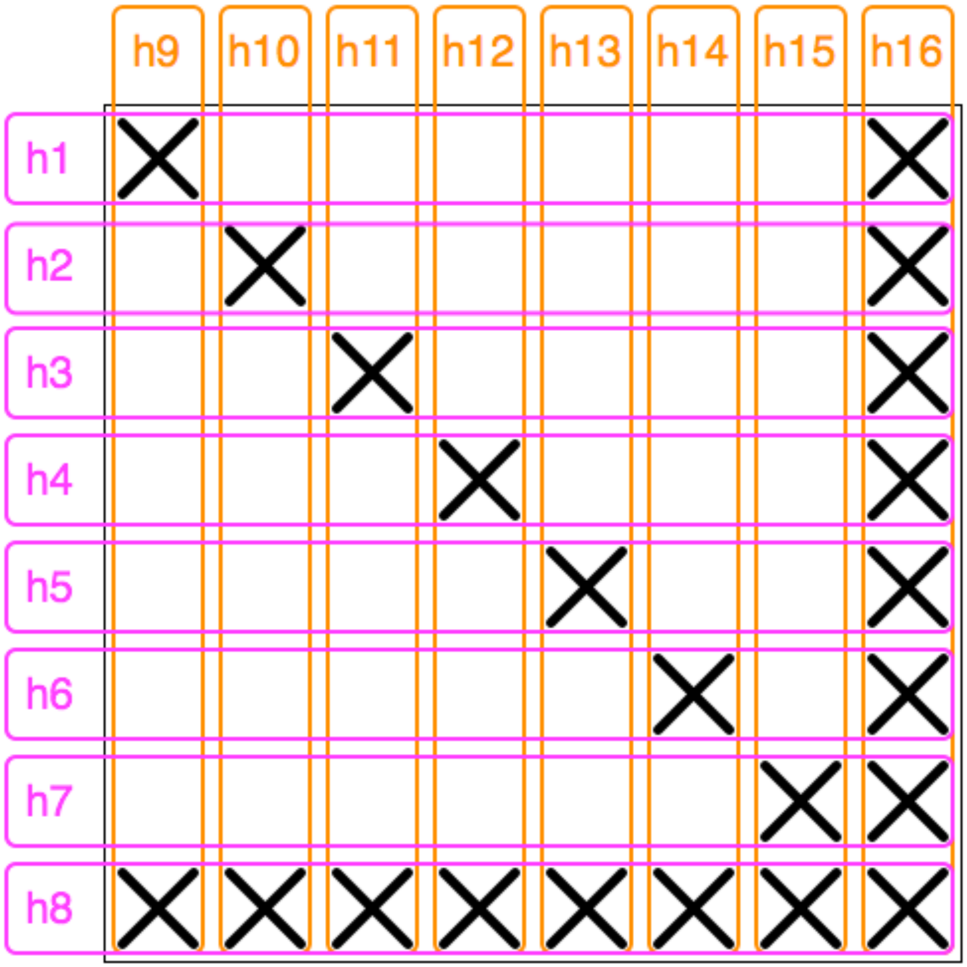
- Rows represented by hyperedges

Fine-Grain Hypergraph Model



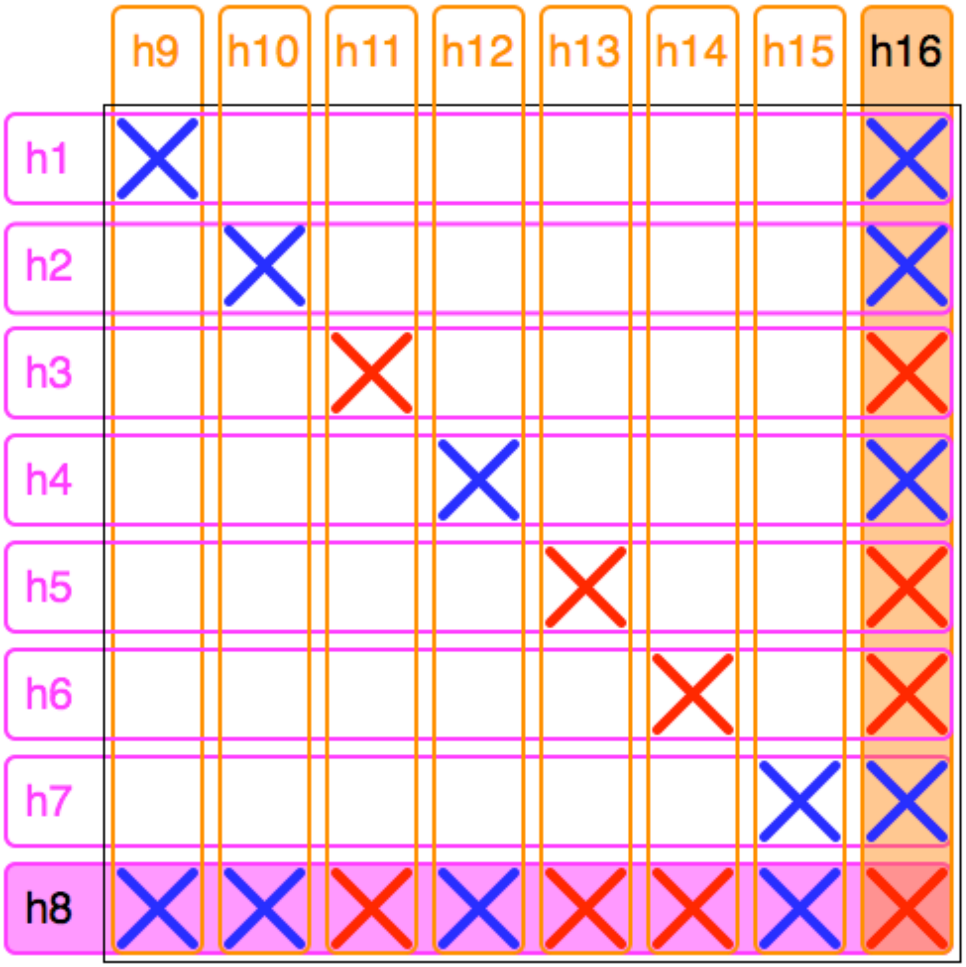
- Columns represented by hyperedges

Fine-Grain Hypergraph Model



- 2n hyperedges

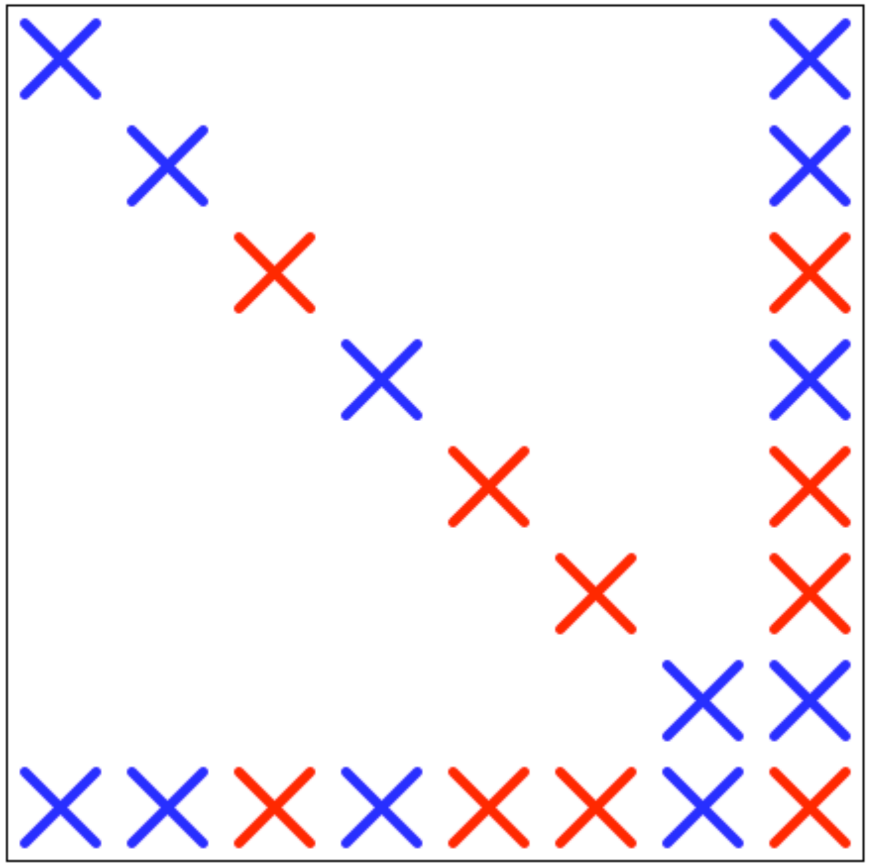
Fine-Grain Hypergraph Model



k=2, volume = cut = 2

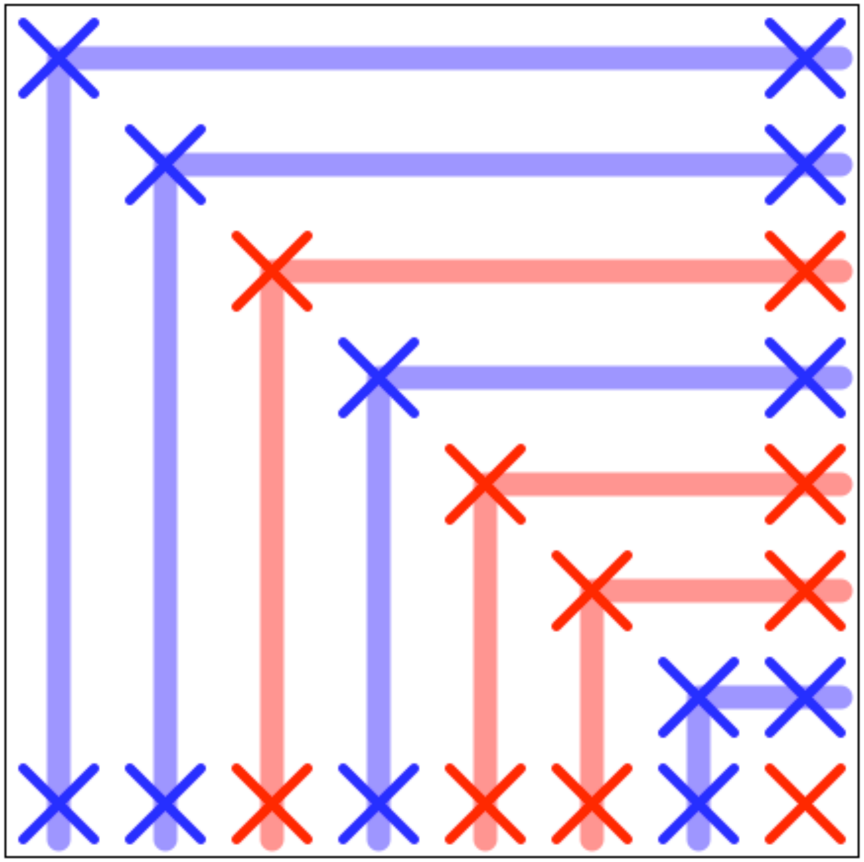
- Partition vertices into k equal sets
- For k=2
  - Volume = number of hyperedges cut
- Minimum volume partitioning when optimally solved
- Larger NP-hard problem than 1-D

New 2-D Method: "Corner" Symmetric Partitioning



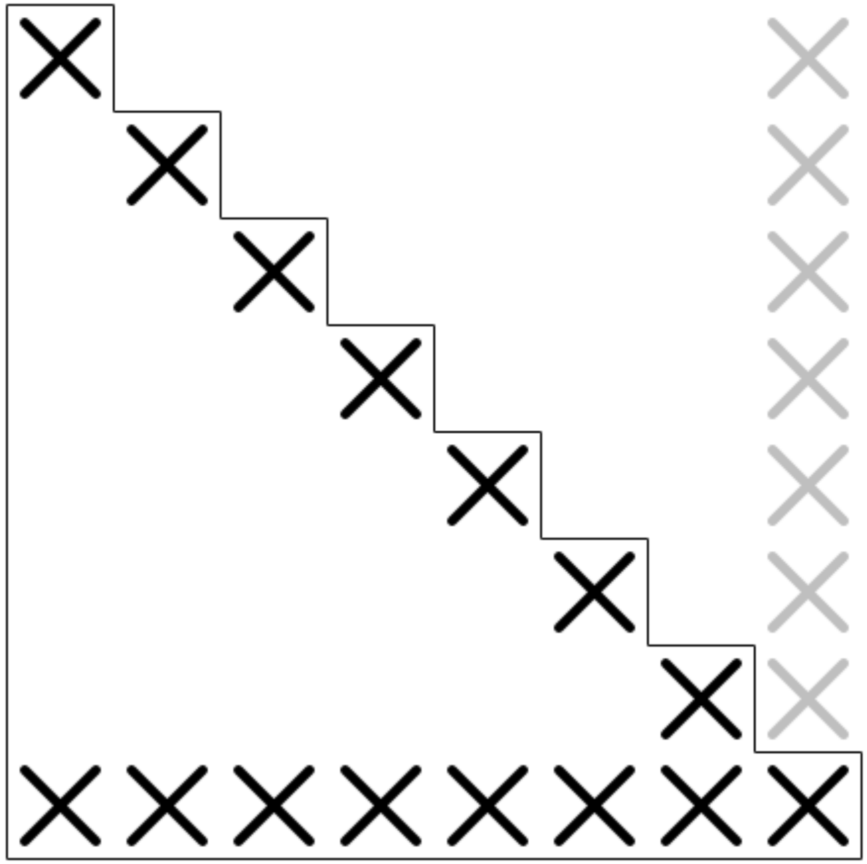
- Optimal partitioning of arrowhead matrix suggests new partitioning method

“Corner” Symmetric Partitioning



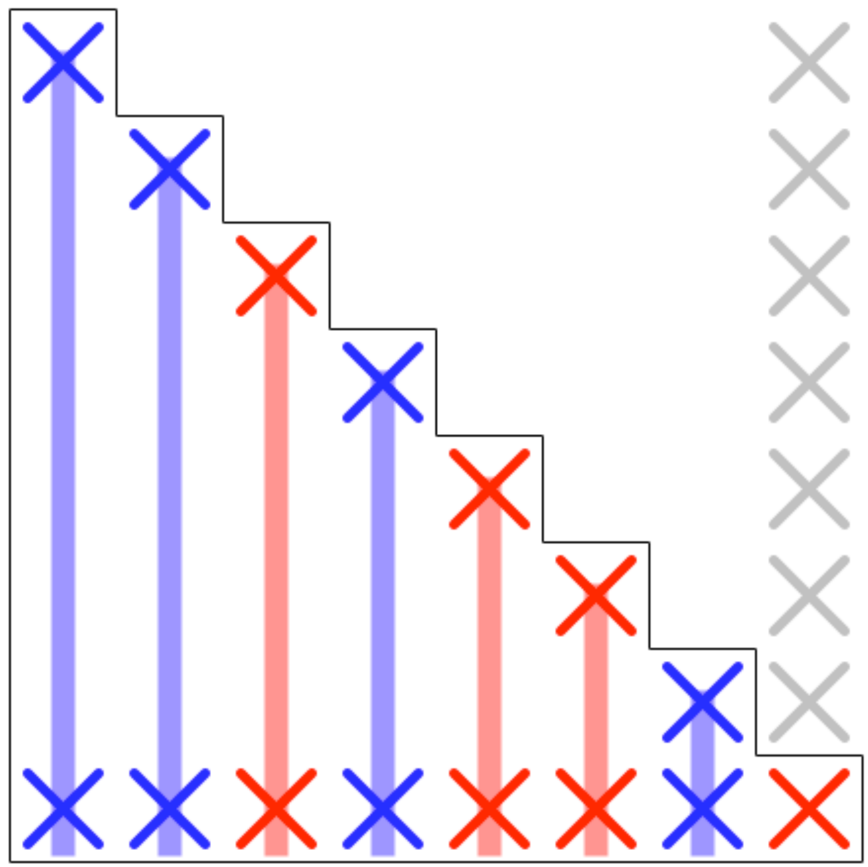
- 1-D parts reflected across diagonal

"Corner" Symmetric Partitioning



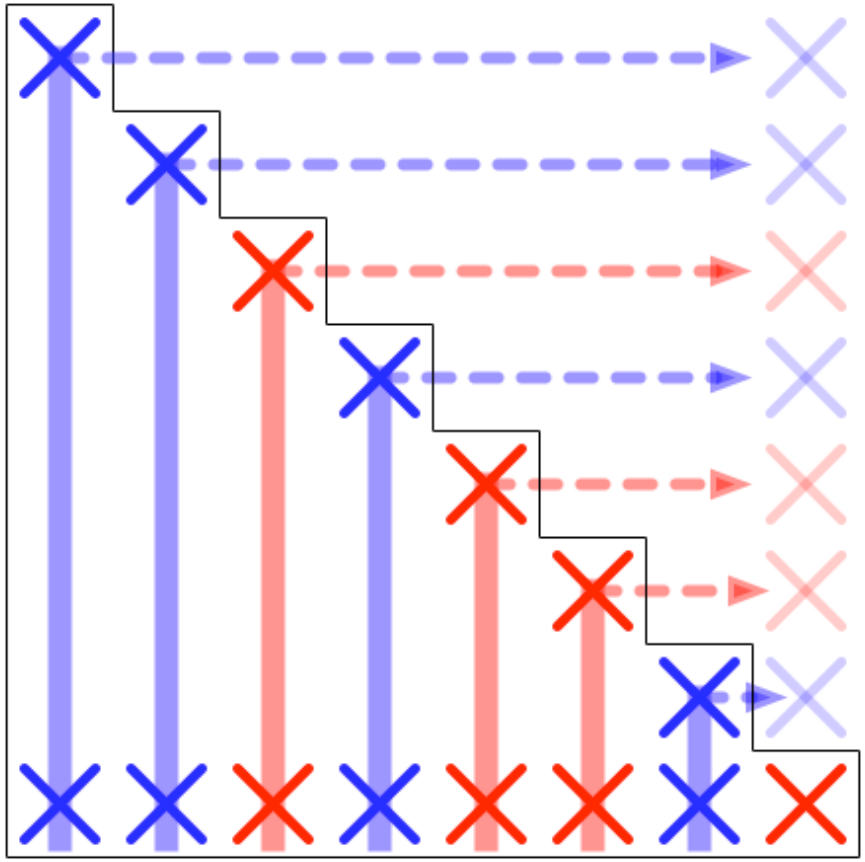
- Take lower triangular portion of matrix

"Corner" Symmetric Partitioning



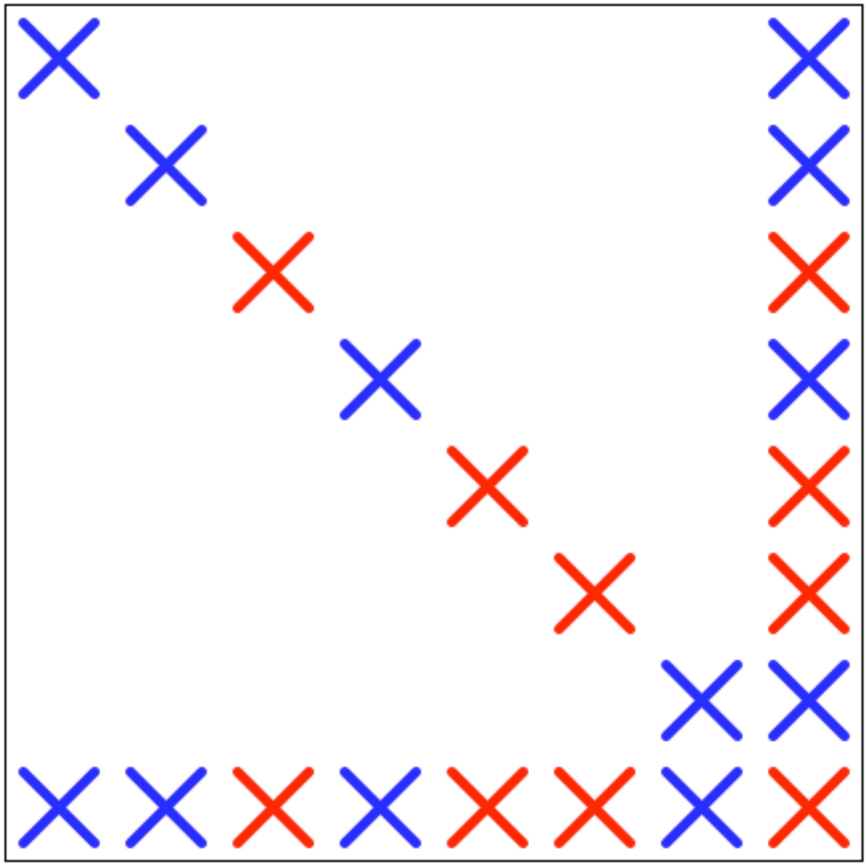
- 1-D (column) hypergraph partitioning of lower triangular matrix

"Corner" Symmetric Partitioning



- Reflect partitioning symmetrically across diagonal

"Corner" Symmetric Partitioning



Volume = 2

- Optimal partitioning

Comparison of Methods -- Arrowhead Matrix

k	1D column	Corner	Fine grain
2	29101	**2***	**2***
4	40001	**6***	**6***
16	40012	**30***	**30***
64	40048	**126***	**126***

Order n

2(k-1)

- n = 40,000
- nnz = 119,998
- Communication volume for 3 methods

*optimal

Preliminary Results

Name	N	nnz	nz/N	$nz/(N)^2$
cage10	11,397	150,645	13.2	1.16×10^{-3}
finan512	74,752	596,992	8.0	1.07×10^{-4}
bcsstk30	28,924	2,043,492	70.7	2.44×10^{-3}
asic680ks	682,712	2,329,176	3.4	5.00×10^{-6}

- Symmetric matrices
- First 3 from Professor Rob Bisseling's (Utrecht University) Mondriaan paper
- Last from Sandia Xyce circuit simulation

- Hypergraph partitioning for all methods
  - Zoltan with PaToH

Preliminary Results: Communication Volume

Name	k	1d hyp.Col	fine-grain hyp.	corner
cage10	2	2308.2	1879.6	**1866.6**
	4	5379.0	**4063.7**	4089.3
	16	12874.5	**8865.5**	8920.9
	64	23463.3	**16334.7**	17164.0
finan512	2	147.8	126.1	**100.0**
	4	295.7	261.2	**215.0**
	16	1216.7	1027.4	**845.0**
	64	9986.0	8624.6	**8135.2**
bcsstk30	2	**605.6**	662.6	618.5
	4	1794.4	1935.7	**1531.0**
	16	8624.7	9774.8	**7232.2**
	64	23308.0	25677.2	**20351.4**
asic680ks	2	1543.5	**686.6**	936.9
	4	3560.4	**1813.3**	2214.2
	16	9998.5	**4634.0**	5562.8
	64	21785.8	**9554.9**	11147.3




Future Work: Reordering

- Ordering not advantageous in 1D methods
  - Same graphs/hypergraph models
- Corner method partitioning quality depends greatly on ordering
  - Ordering impacts off-diagonal nz partitioning

- Symmetric reordering to further reduce communication
- Focus on bisection
  - Recursive bisection for k>2

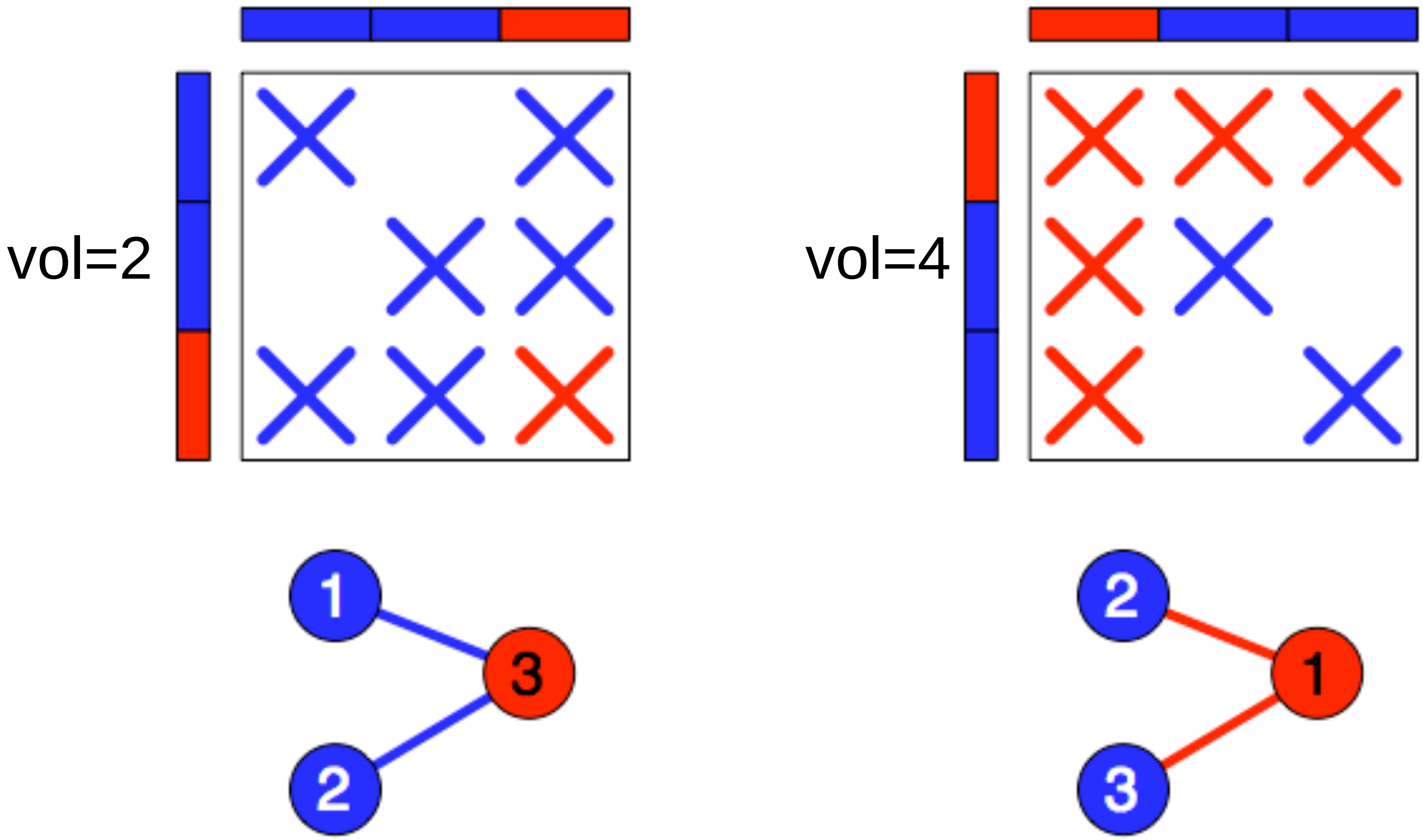
Reordering (Bisection)

- Graph model G(V,E)
  - Vector entries represented by vertices
  - Off-diagonal nonzeros represented by edges
  - Each vertex v_i assigned part s_i and position π_i
- v_i "costs" 2 words of communication iff

$$\exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j$$

- v_i "free" otherwise

Reordering (Bisection)



- v_i "costs" 2 words if

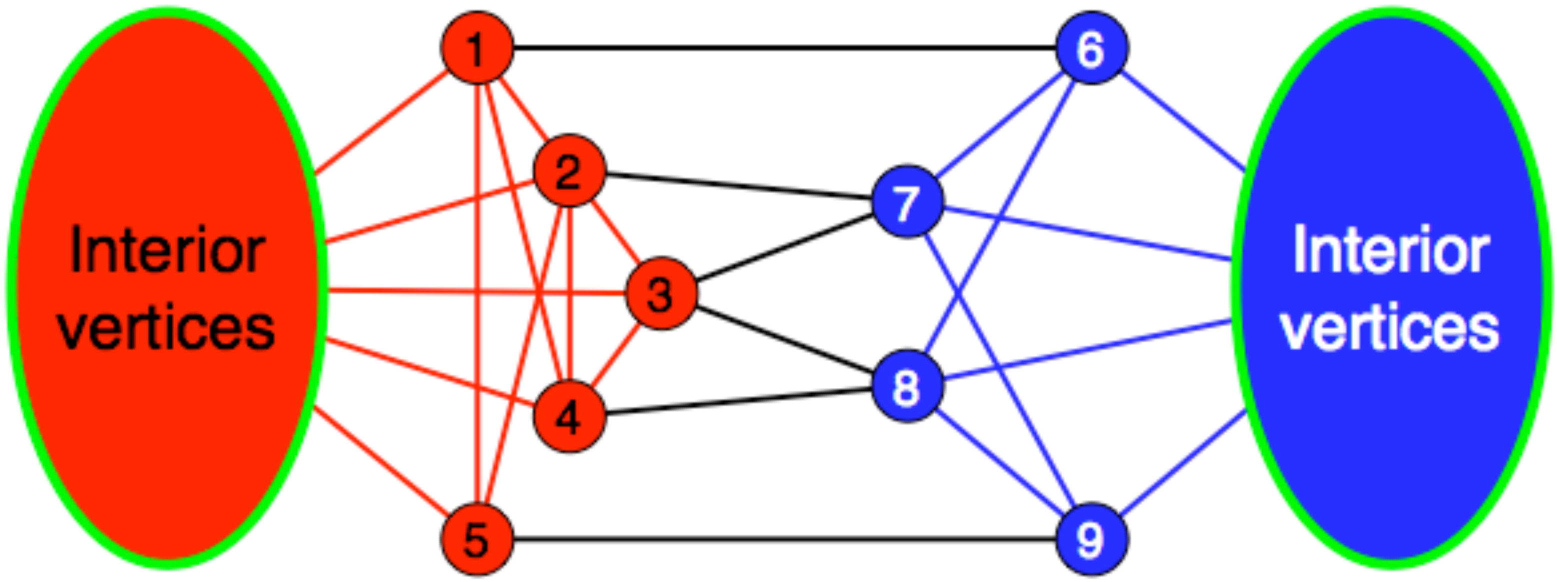
$$\exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j$$

Reordering (Bisection)

- Ideally find optimal partitioning/ordering
  - Very difficult combinatorial problem
- Instead we propose
  - Fix ordering, partition
    - Corner method
  - Fix vertex partitioning, find optimal ordering
- Can iterate two steps

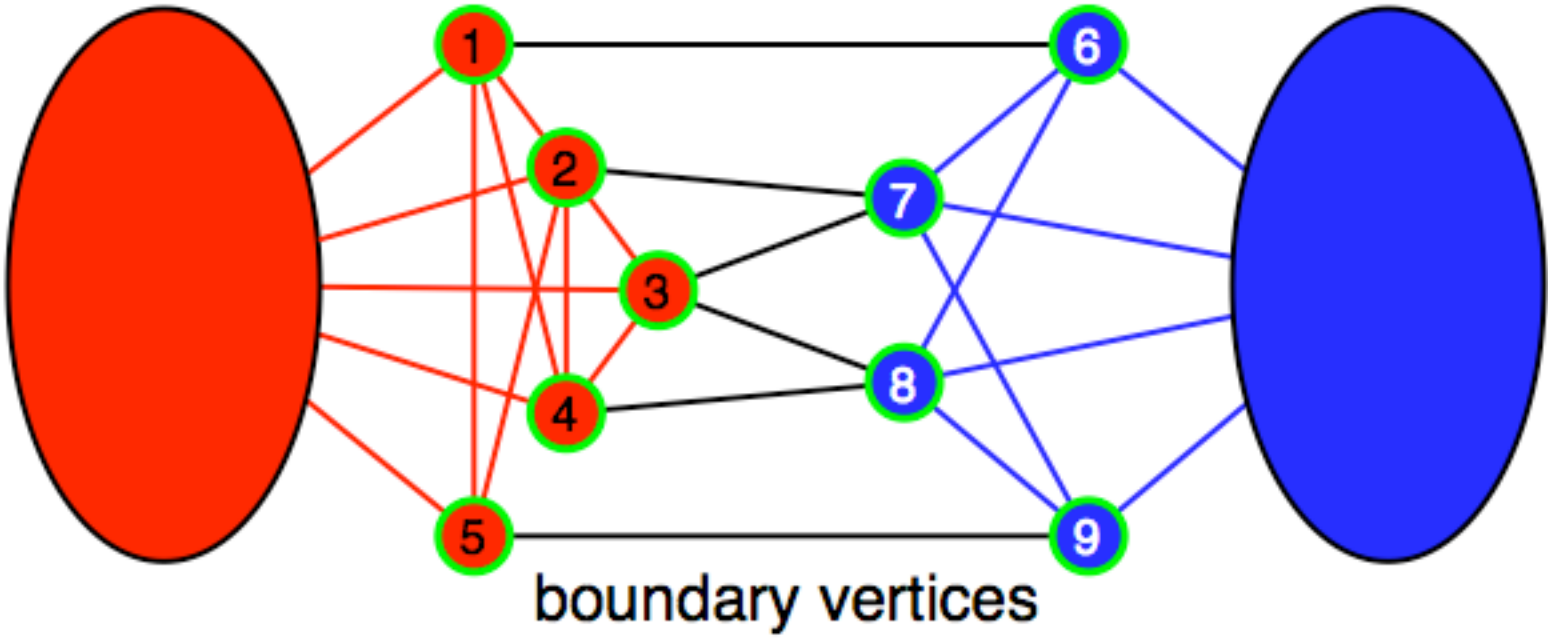
- Need to find optimal vertex ordering given fixed vertex partitioning
  - Divide graph into 3 categories of vertices

Reordering (Bisection): Vertex Categories



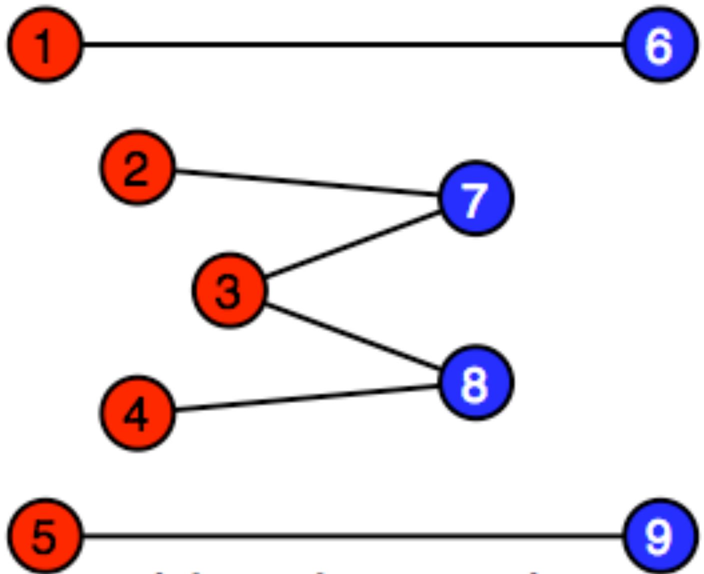
- Interior vertices: not adjacent to any vertex owned by different part

Reordering (Bisection): Vertex Categories



- Boundary vertices: adjacent to at least one vertex owned by different part

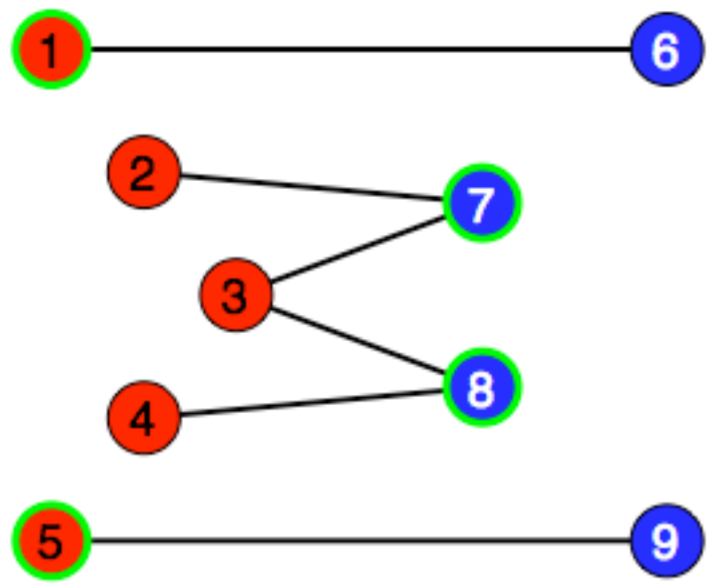
Reordering (Bisection): Vertex Categories



bipartite graph

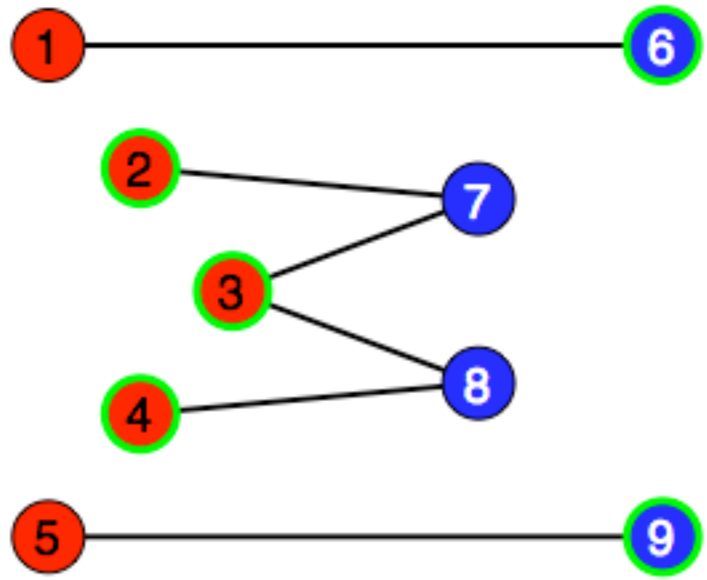
- Bipartite graph obtained by
  - Removing interior vertices
  - Removing non-cut edges

Reordering (Bisection): Vertex Categories



- Minimum vertex cover of bipartite graph
  - Cover boundary vertices

Reordering (Bisection): Vertex Categories



- Non-cover boundary vertices

Reordering (Bisection): Vertex Categories

- 3 Categories
  - Interior vertices
  - Non-cover boundary vertices
  - Cover boundary vertices

Reordering (Bisection): Ordering Interior V

- v_i "costs" if

$$\exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j$$

- Interior vertices can be given any position with no affect on volume
  - Since adjacent vertices have same part
  - Position these first

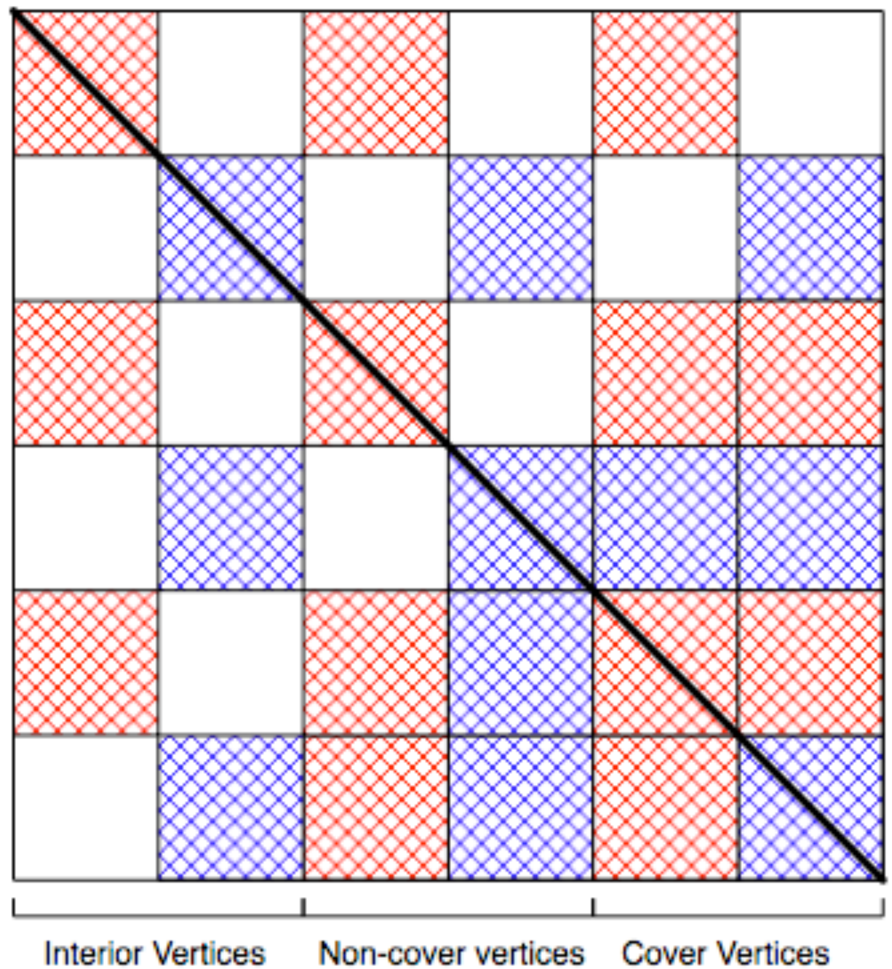
Reordering (Bisection): Ordering Other V

- v_i "costs" if

  $$\exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j$$

- Find ordering of remaining V such that
  - Minimum set of vertices result in communication
  - Equivalently, minimum set of vertices such that for each edge in bipartite graph, vertex with larger numbered position is contained in this set
  - Minimum vertex cover gives us this set
    - With cover vertices ordered last
- Order cover boundary vertices last

Reordering (Bisection): Resulting Matrix

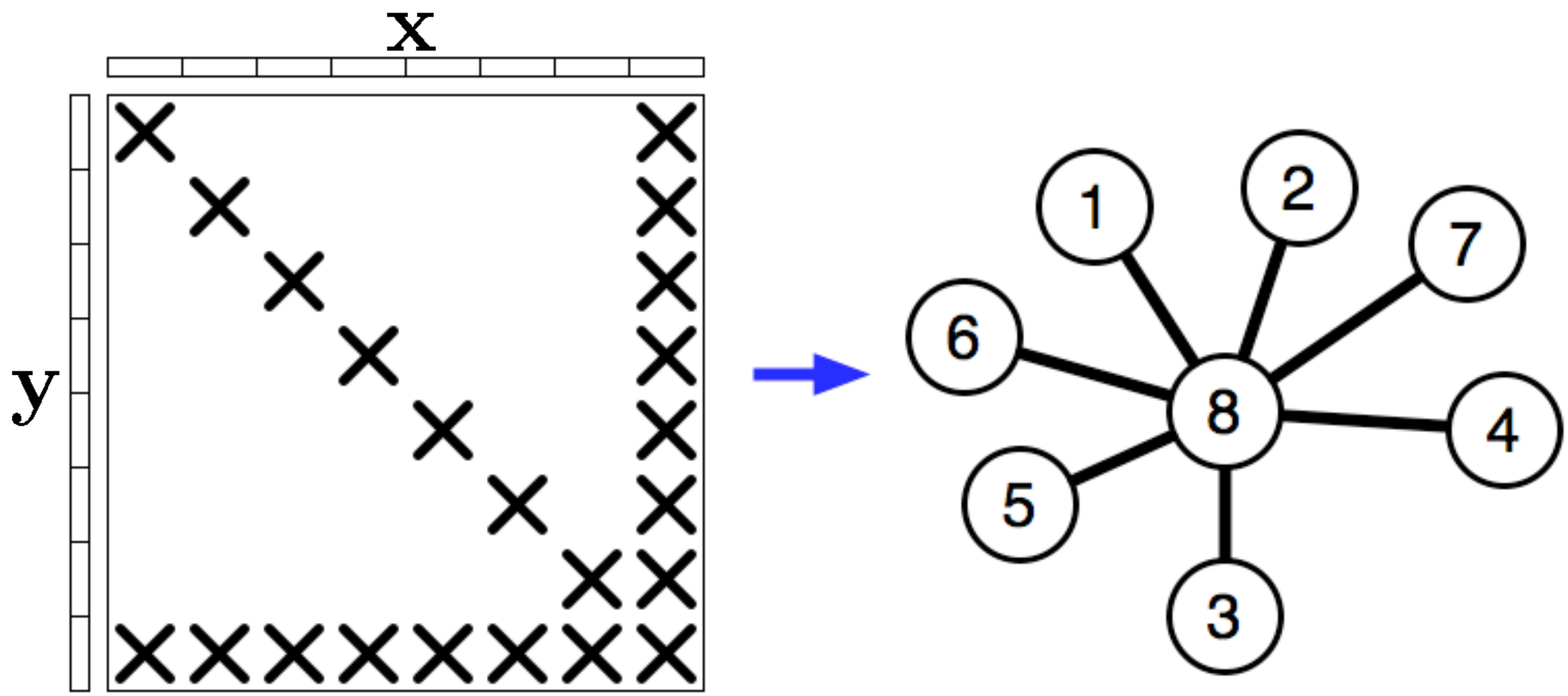


- Only cover boundary vertices “cost”

Graph Model for Symmetric 2-D Partitioning

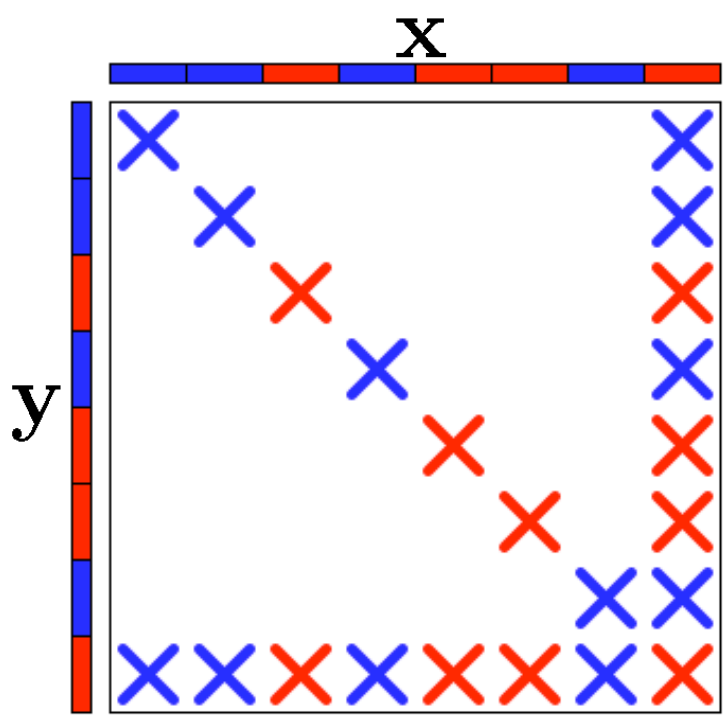
- Given symmetric matrix A
- Symmetric partition
  - a(i,j) and a(j,i) assigned same partition
  - Input and output vectors have same distribution
- Corresponding graph G(V,E)
  - Vertices correspond to vector elements
  - Edges correspond to off-diagonal nonzeros

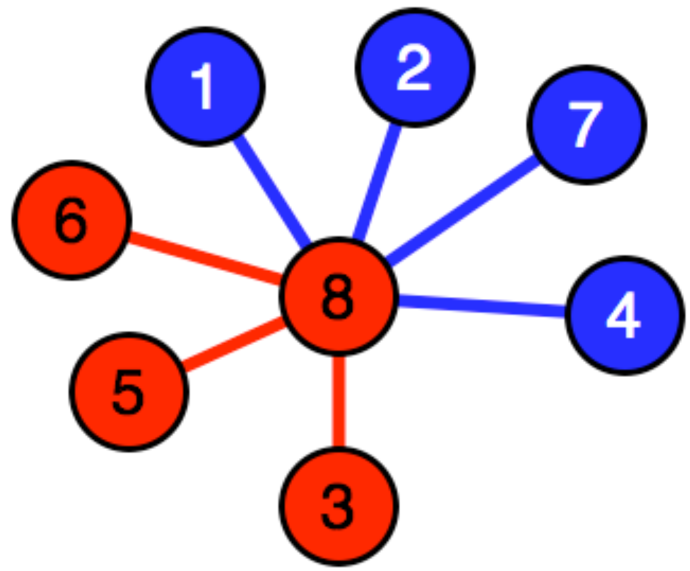
Graph Model for Symmetric 2-D Partitioning



- Corresponding graph G(V,E)
  - Vertices correspond to vector elements
  - Edges correspond to off-diagonal nonzeros

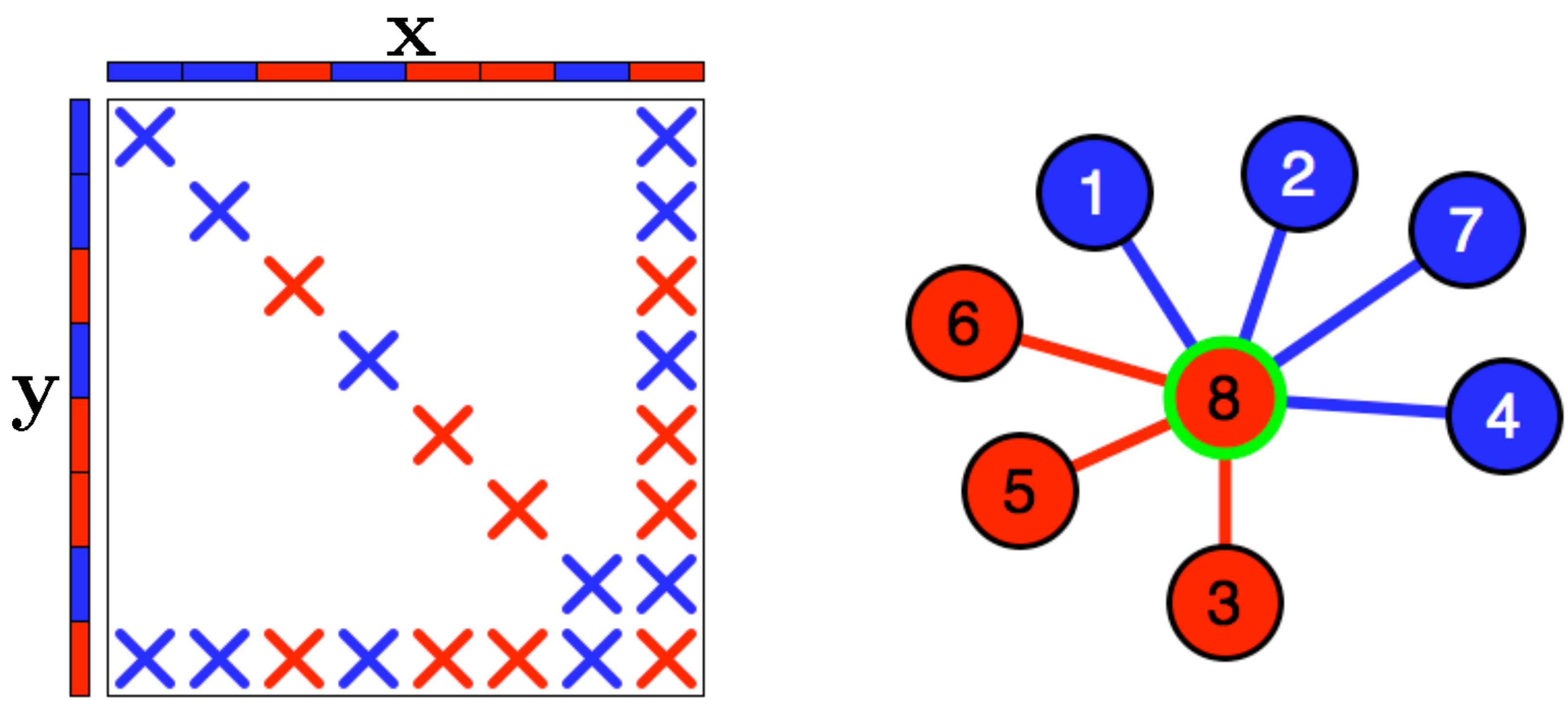
Graph Model for Symmetric 2-D Partitioning





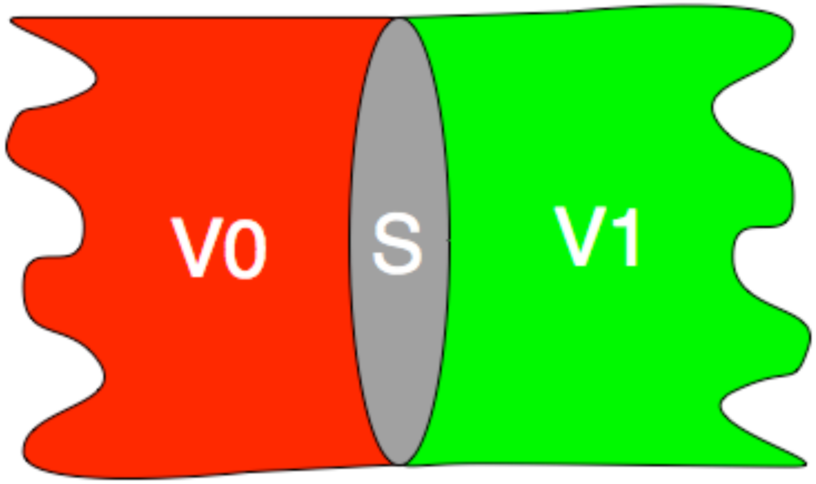
- Symmetric 2-D partitioning
  - Partition both V and E
  - Gives partitioning of both matrix and vectors

Communication in Graph Model



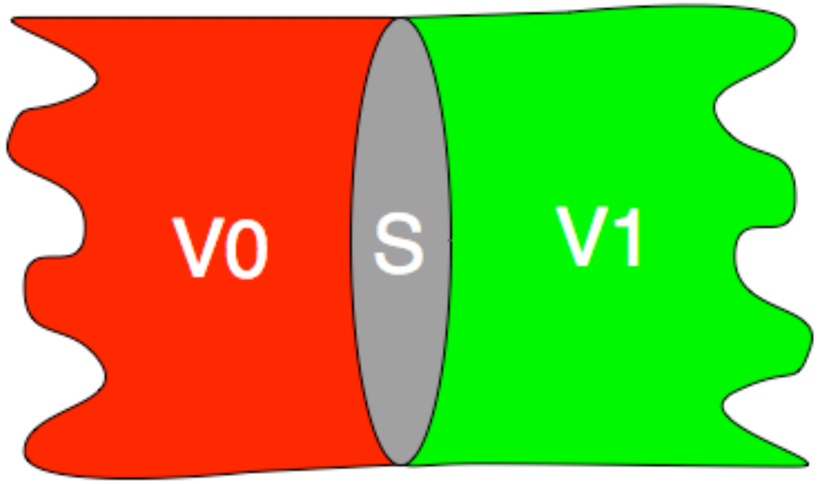
- Communication is assigned to vertices
- Vertex incurs communication iff incident edge is in different part
- Want small vertex separator -- $S=\{V_8\}$

Nested Dissection Partitioning Method - Bisection



- Suppose A is symmetric
- Let G(V,E) be graph of A
- Find small, balanced separator S
  - Yields vertex partitioning V = (V0,V1,S)
- Partition the edges
  - E0 = {edges that touch a vertex in V0}
  - E1 = {edges that touch a vertex in V1}

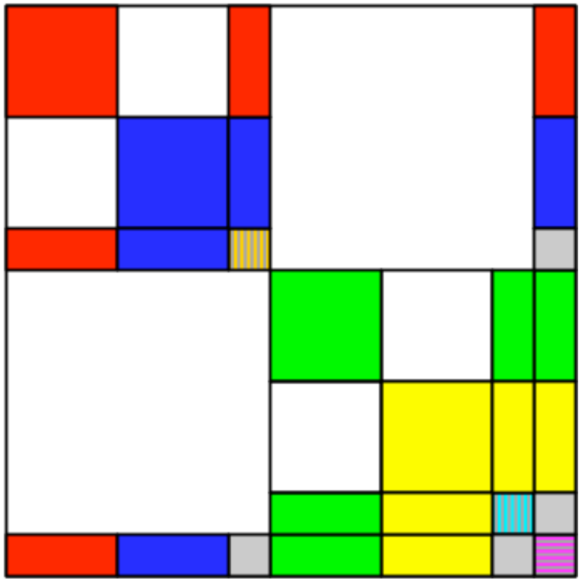
Nested Dissection Partitioning Method - Bisection

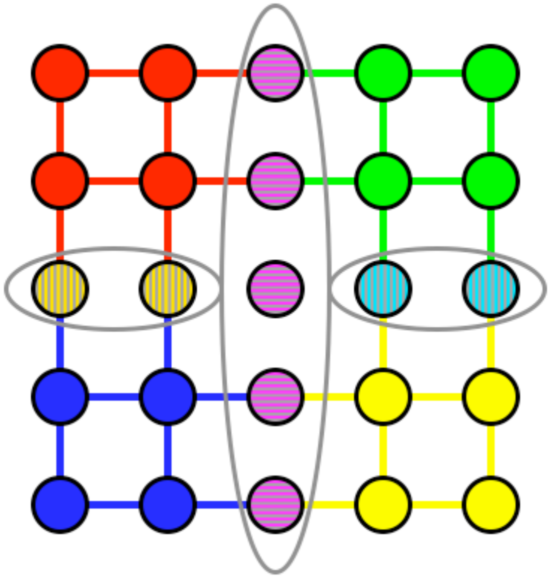


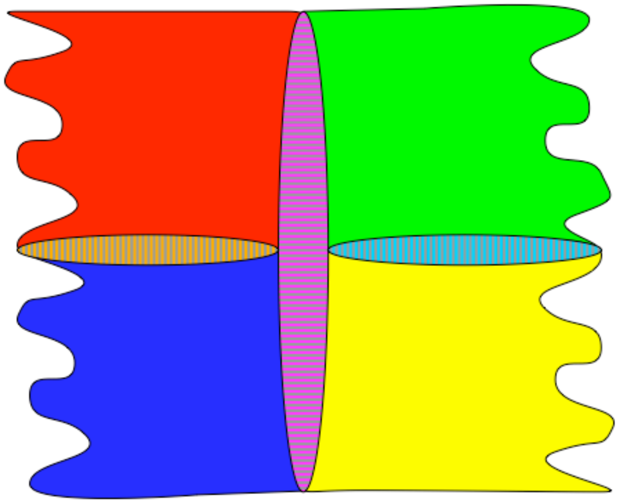
- Vertices in S and corresponding edges
  - Can be assigned to either partition
  - Can use flexibility to maintain balance
- Communication Volume = 2*|S|
  - Regardless of S partitioning
  - |S| in each phase

Nested Dissection Partitioning Method

- Recursive bisection to partition into >2 partitions
- Use nested dissection!



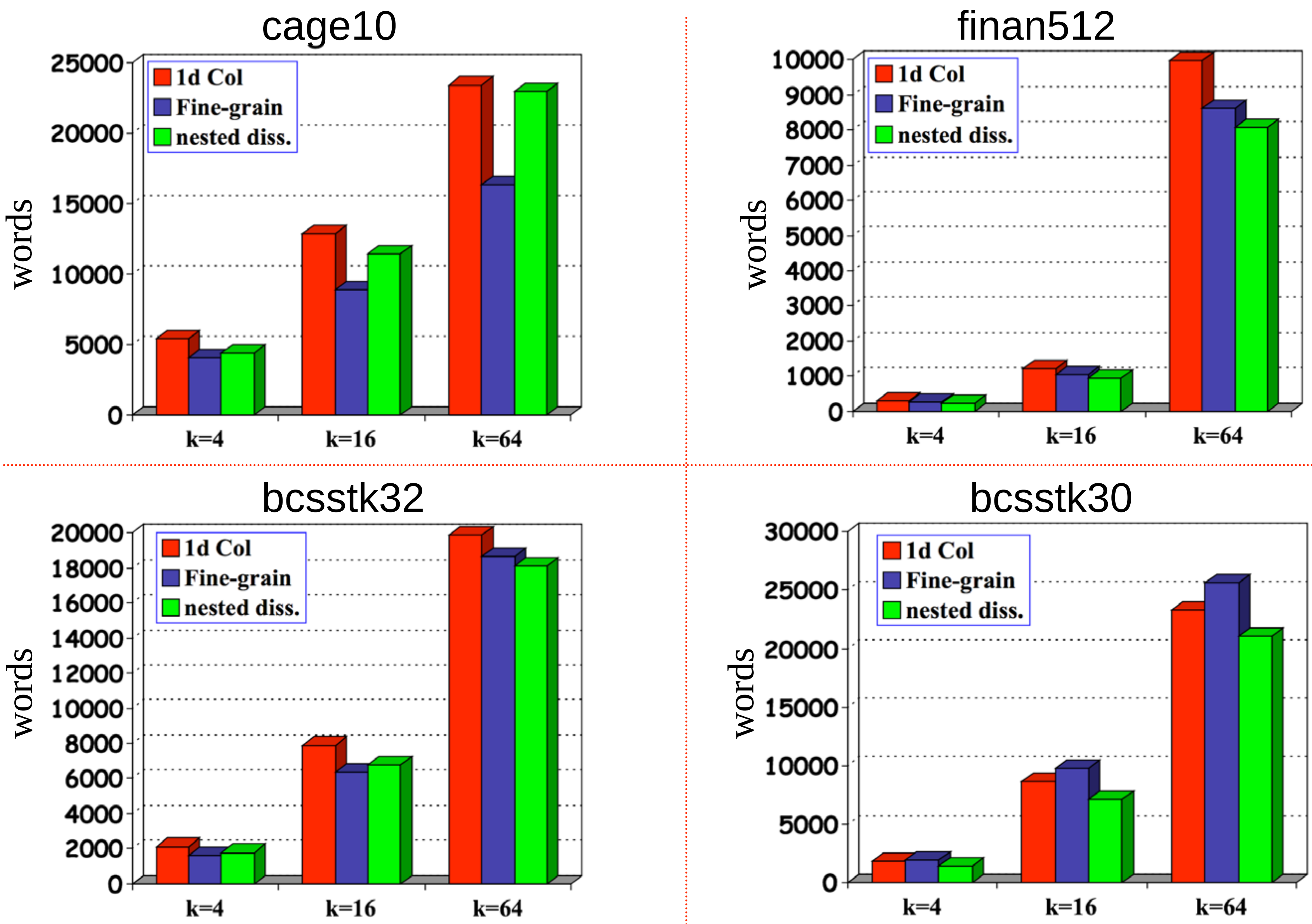




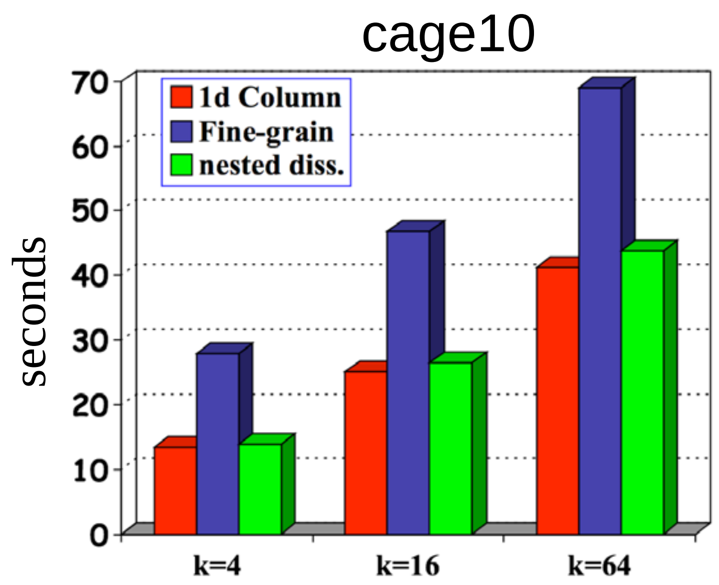
Preliminary Numerical Experiments

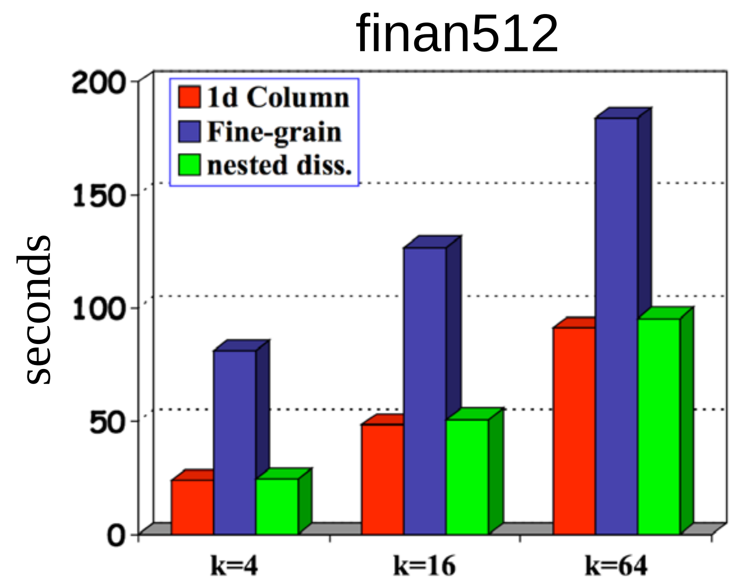
- Compared 3 methods
  - 1-D hypergraph partitioning
  - Fine-grain hypergraph partitioning
  - Nested dissection partitioning
- Hypergraph partitioning for all methods
  - Zoltan with PaToH
- Symmetric and nonsymmetric matrices
  - Mostly from Prof. Rob Bisseling (Utrecht Univ.)
- k = 4, 16, 64 partitions

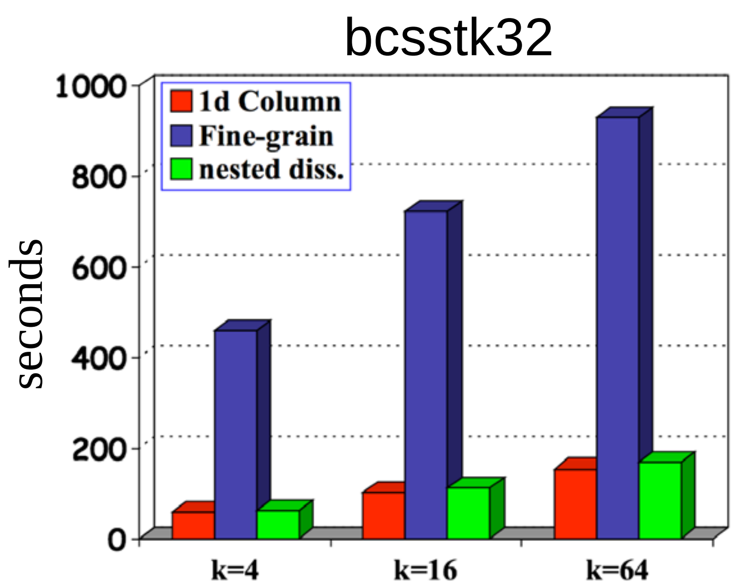
Communication Volume - Symmetric Matrices

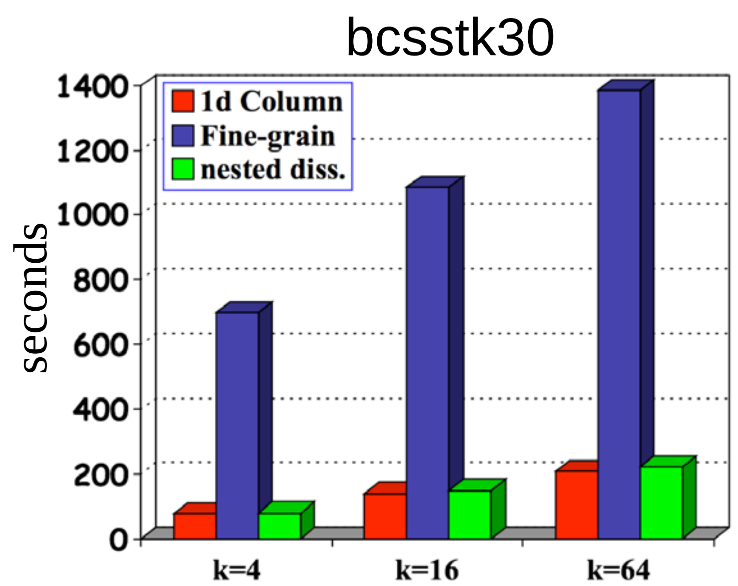


Runtimes


cage10
seconds
70
60
50
40
30
20
10
0
1d Column
Fine-grain
nested diss.
k=4
k=16
k=64

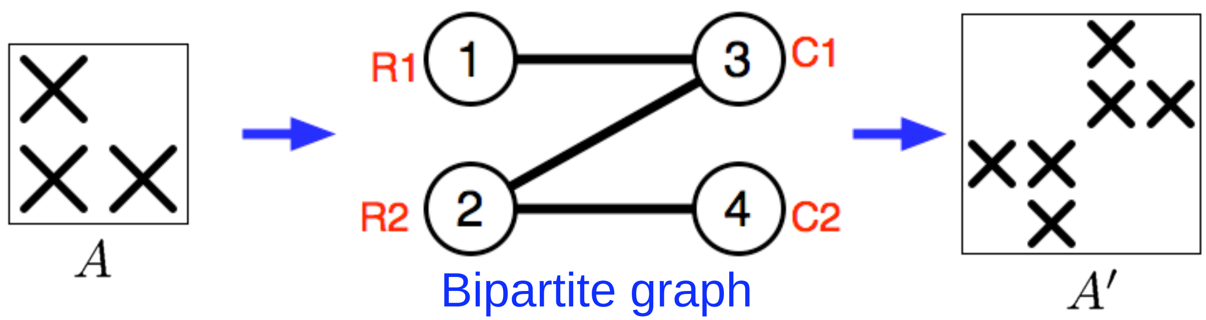

finan512
seconds
200
150
100
50
0
1d Column
Fine-grain
nested diss.
k=4
k=16
k=64


bcsstk32
seconds
1000
800
600
400
200
0
1d Column
Fine-grain
nested diss.
k=4
k=16
k=64


bcsstk30
seconds
1400
1200
1000
800
600
400
200
0
1d Column
Fine-grain
nested diss.
k=4
k=16
k=64

Nonsymmetric Matrices

- Given nonsymmetric matrix A
- Construct bipartite graph $G'(R,C,E)$
  - R vertices correspond to rows, C vertices to columns
  - E correspond to nonzeros
  - Can be represented by symmetric adjacency matrix



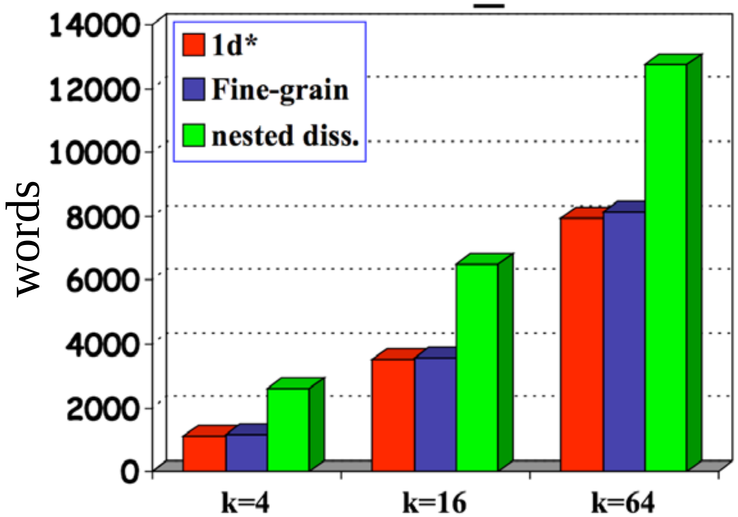
$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$

- Apply nested dissection approach to G'
  - Use same algorithm as for symmetric case

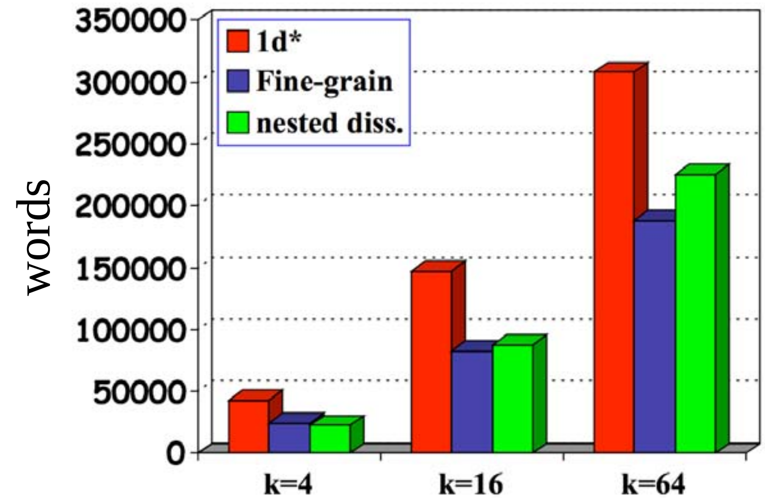
Communication Volume - Nonsymmetric Matrices

Rectangular

cre_b

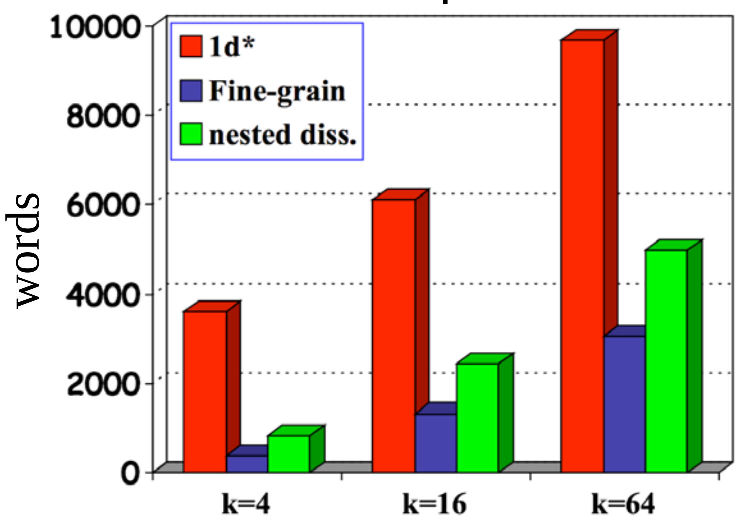


tbdlinux

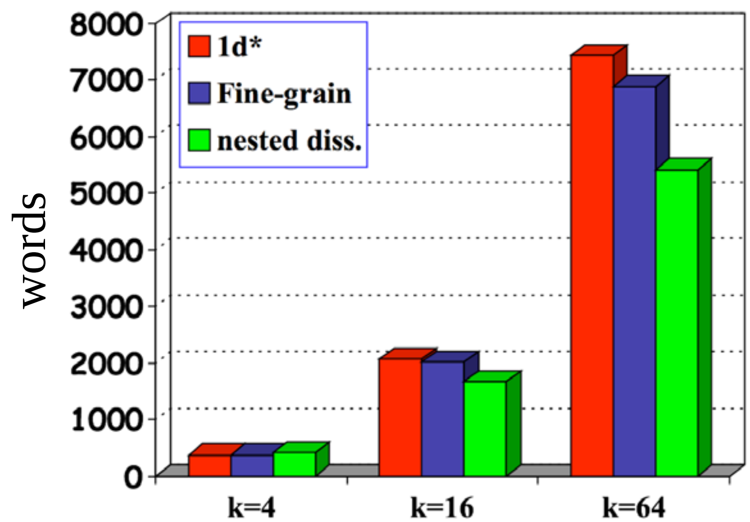


Square

memplus

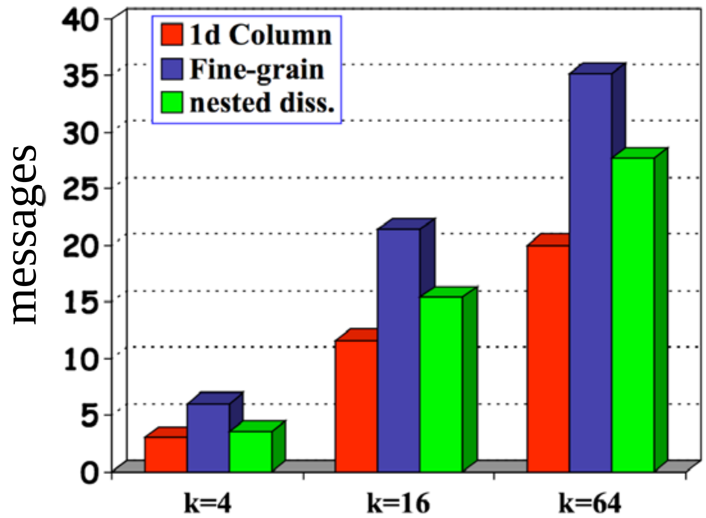


lhr34

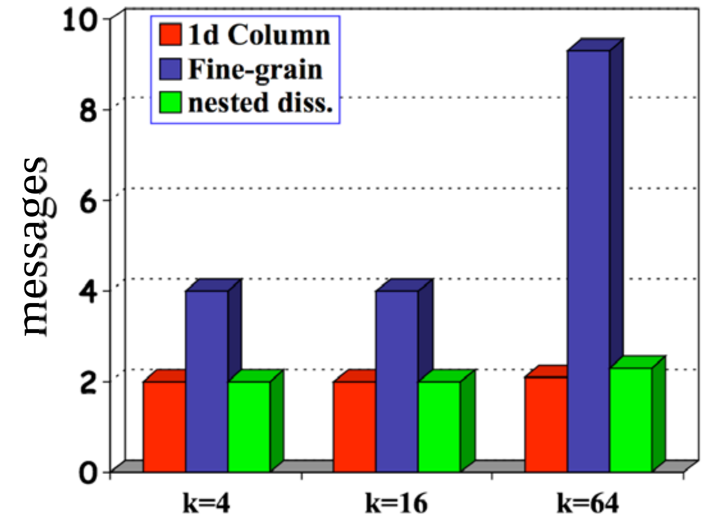


Messages Sent (or Received) per Process

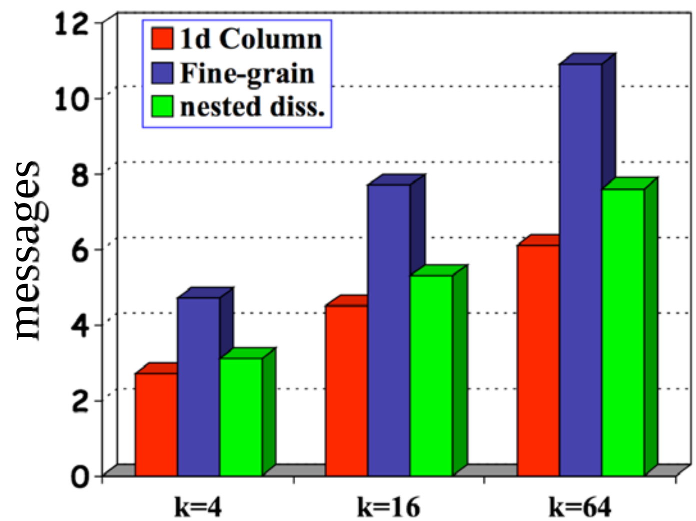
cage10



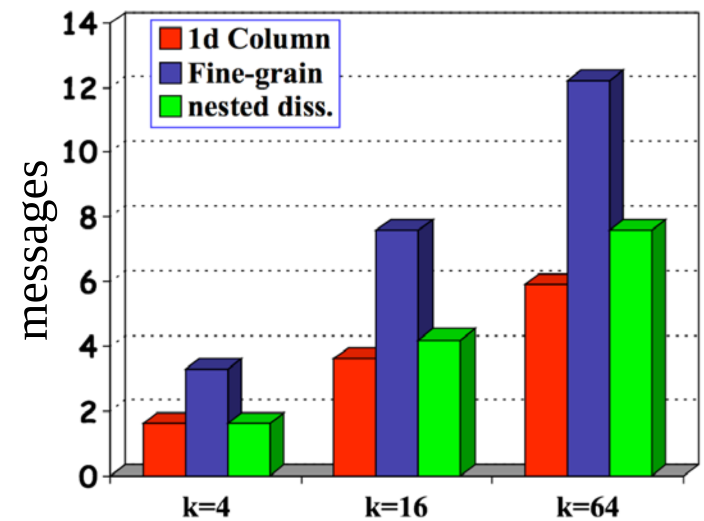
finan512



bcsstk32



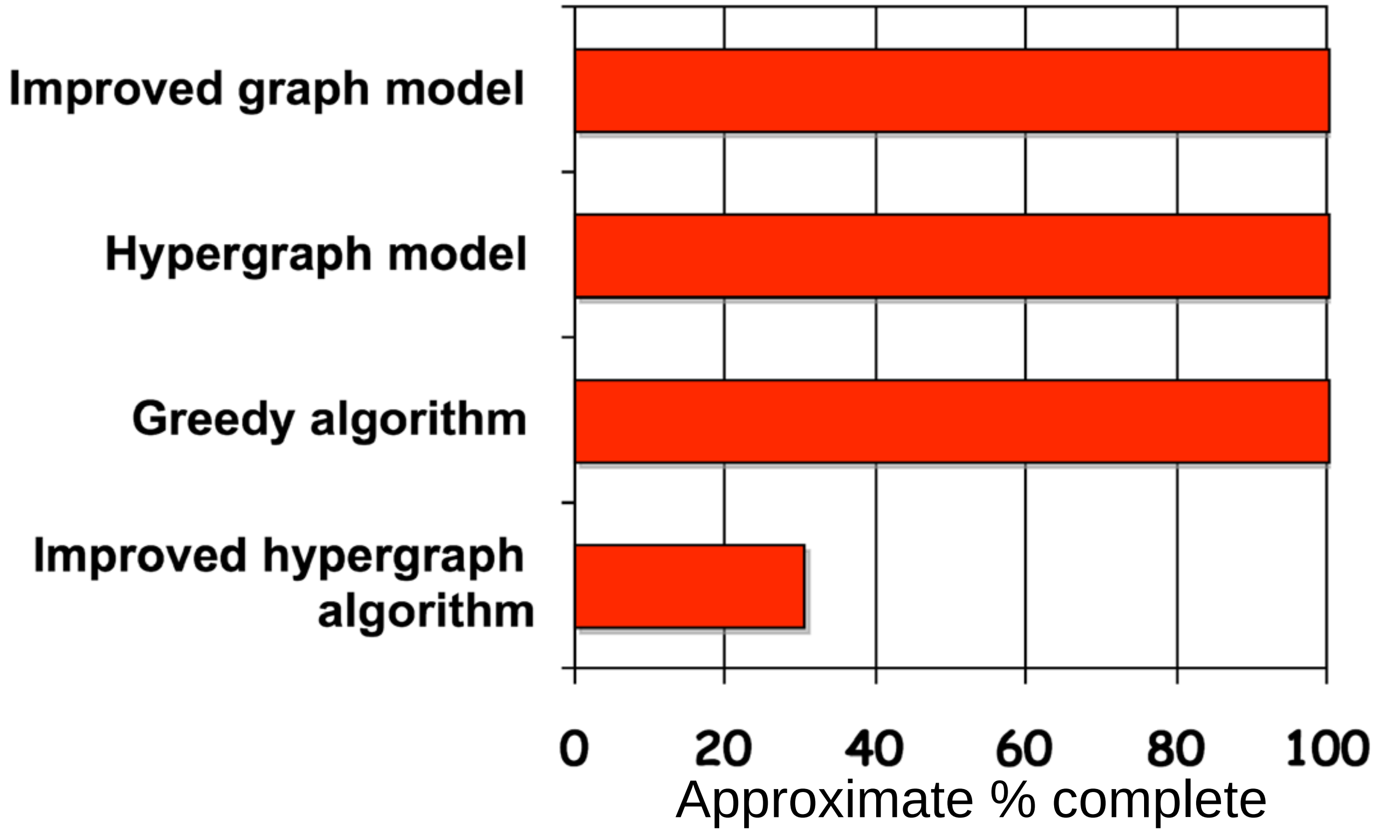
bcsstk30



Summary of Nested Dissection Method Results

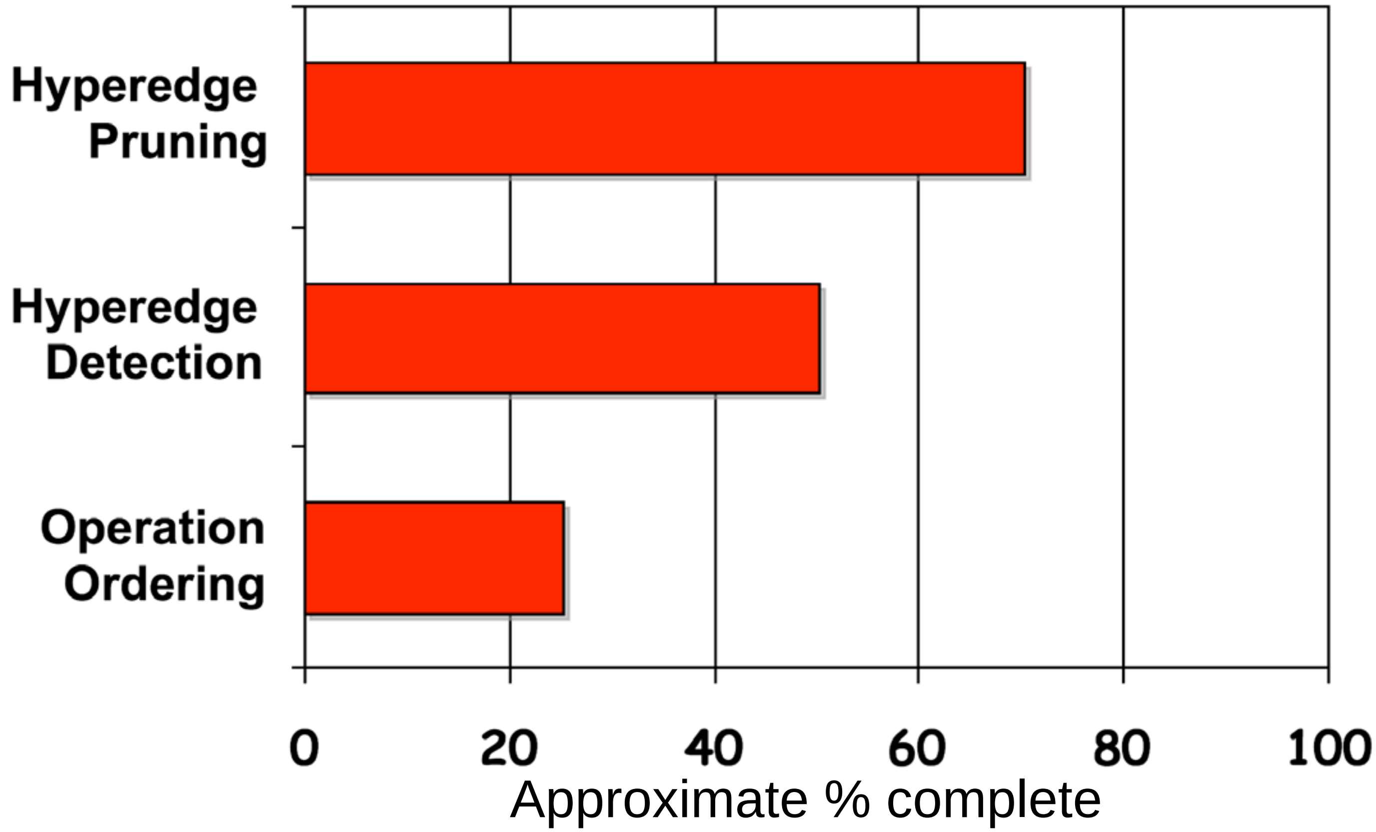
- New nested dissection 2-D algorithm
  - Implemented using existing algorithms and software
  - Quality better than 1-D, and similar to fine-grain hypergraph method for many matrices
  - Faster to compute than fine-grain hypergraph
  - Fewer messages than fine-grain hypergraph

Progress: Serial Matrix-Vector Multiplication (1)



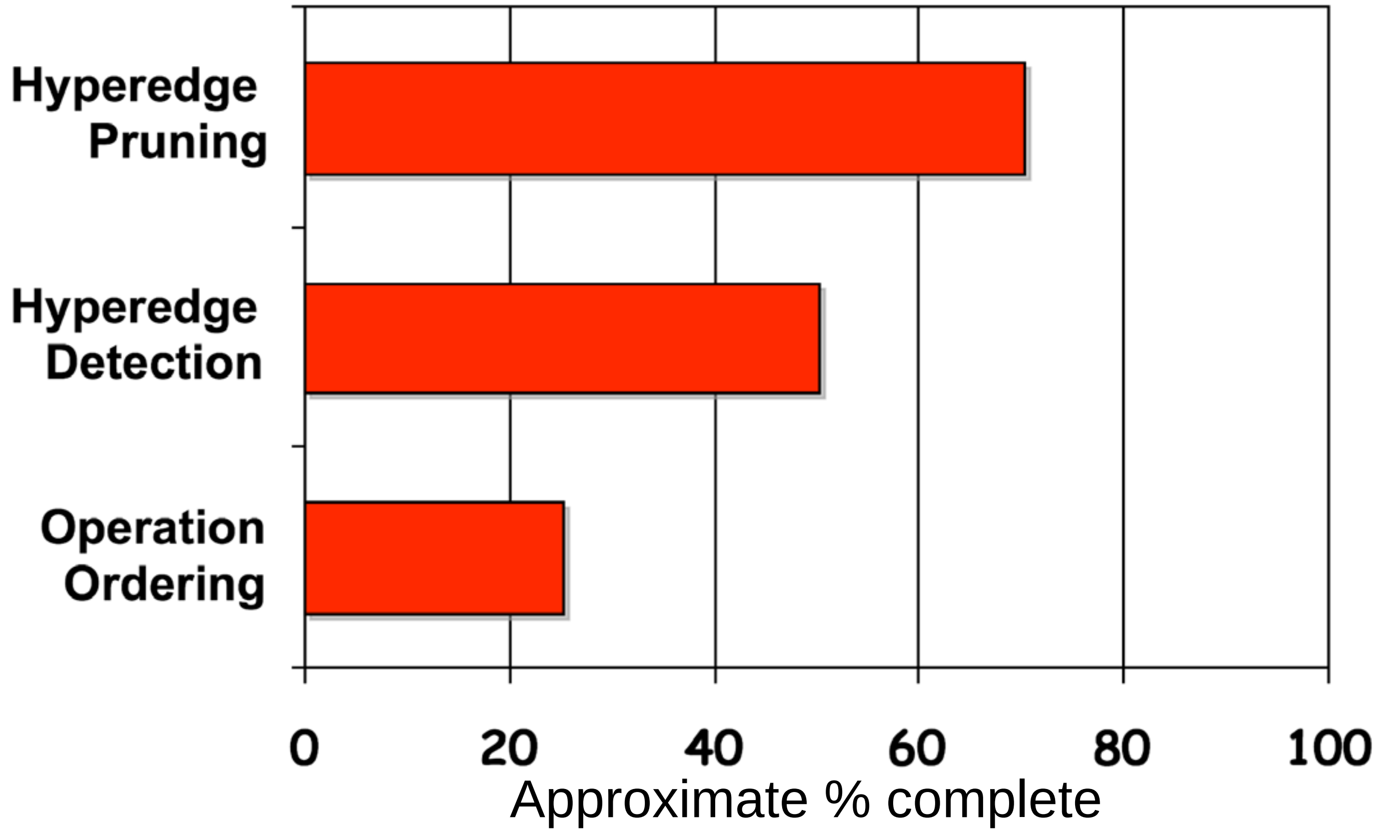
- Improved hypergraph algorithm
  - Develop vertex ordering algorithm

Progress: Serial Matrix-Vector Multiplication (2)



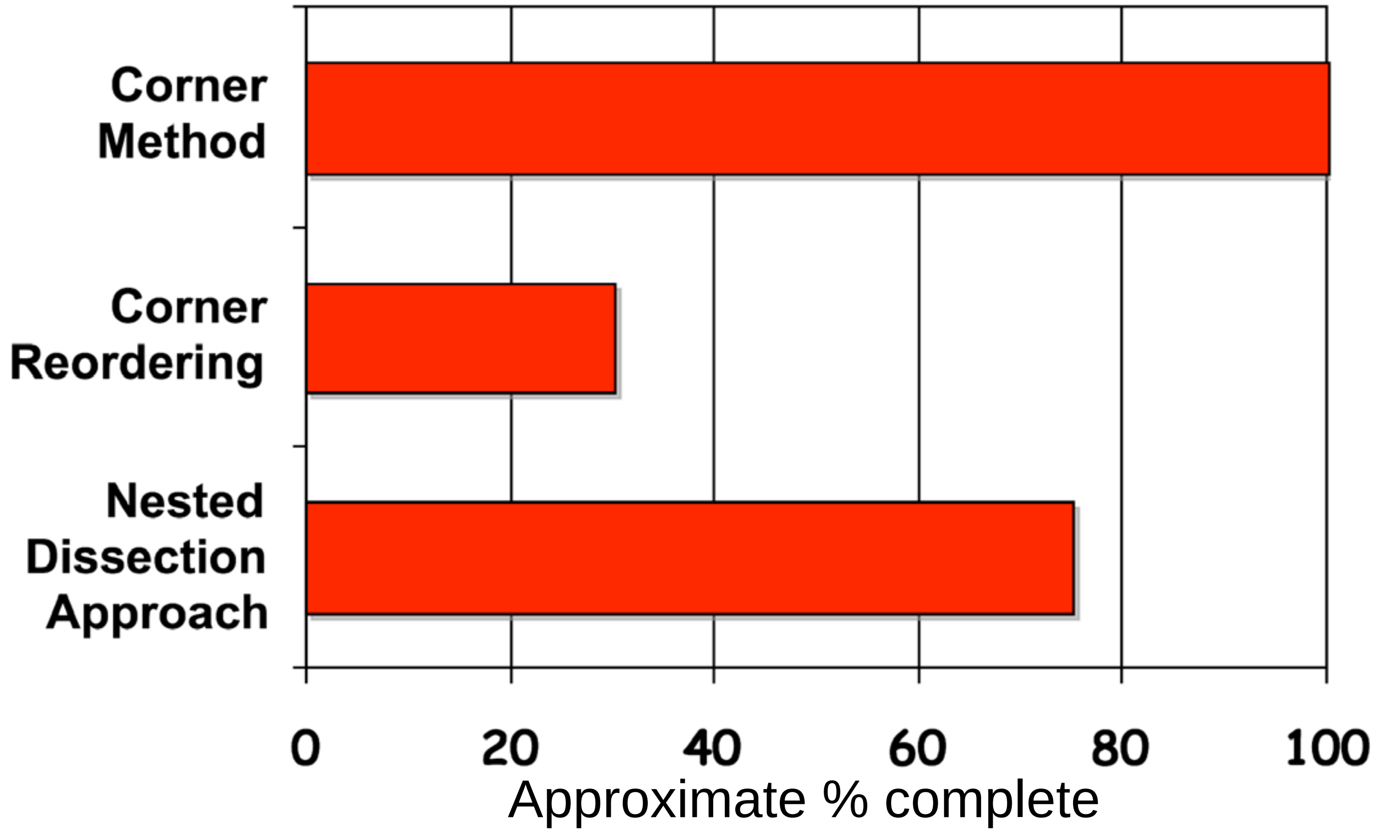
- Hyperedge pruning
  - Develop one or two more heuristics
  - One based on MST graph solution

Progress: Serial Matrix-Vector Multiplication (2)



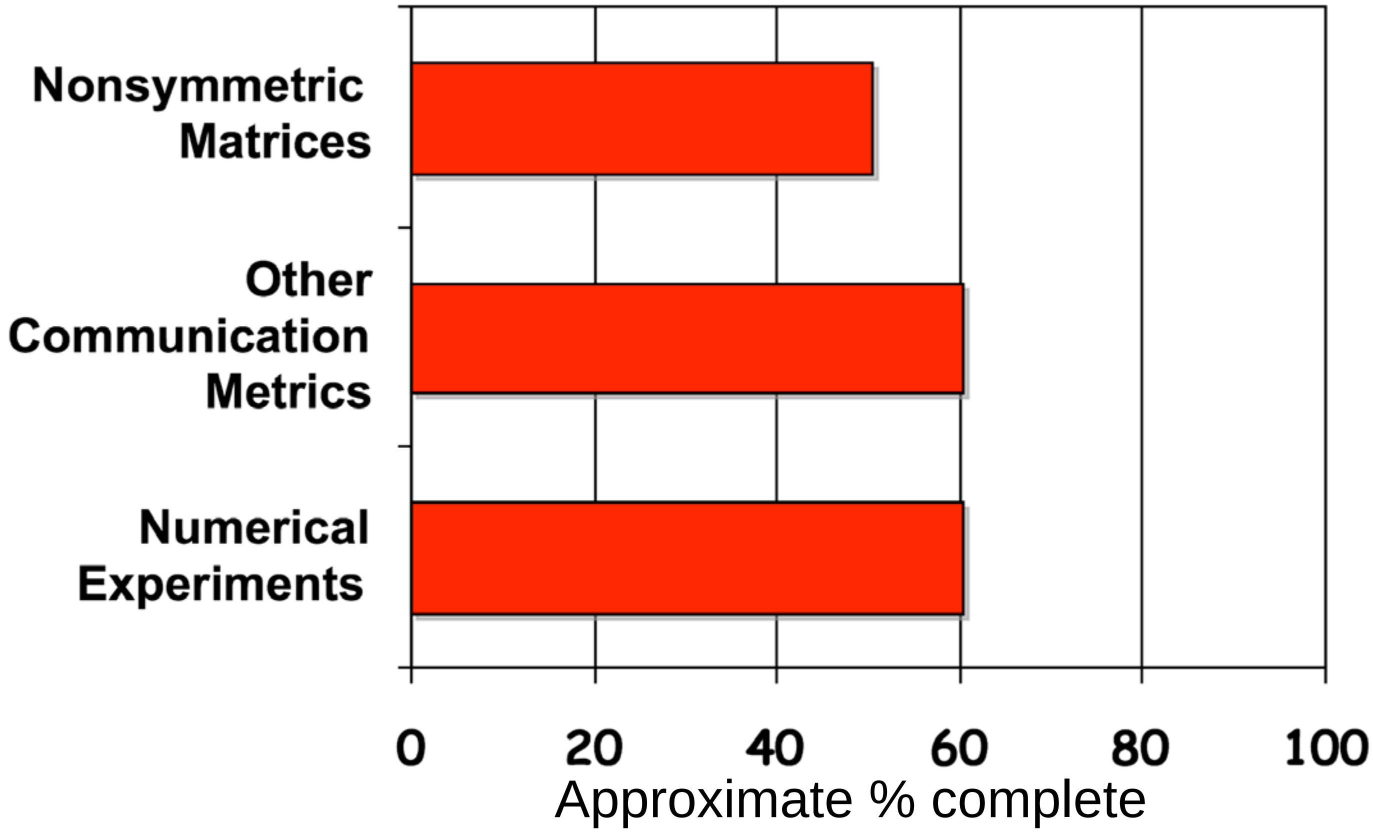
- Hyperedge detection
  - Need to improve $O(n^3)$ looping (for coplanar)
- Operation ordering
  - More cache friendly ordering

Progress: Sparse Matrix Partitioning (1)



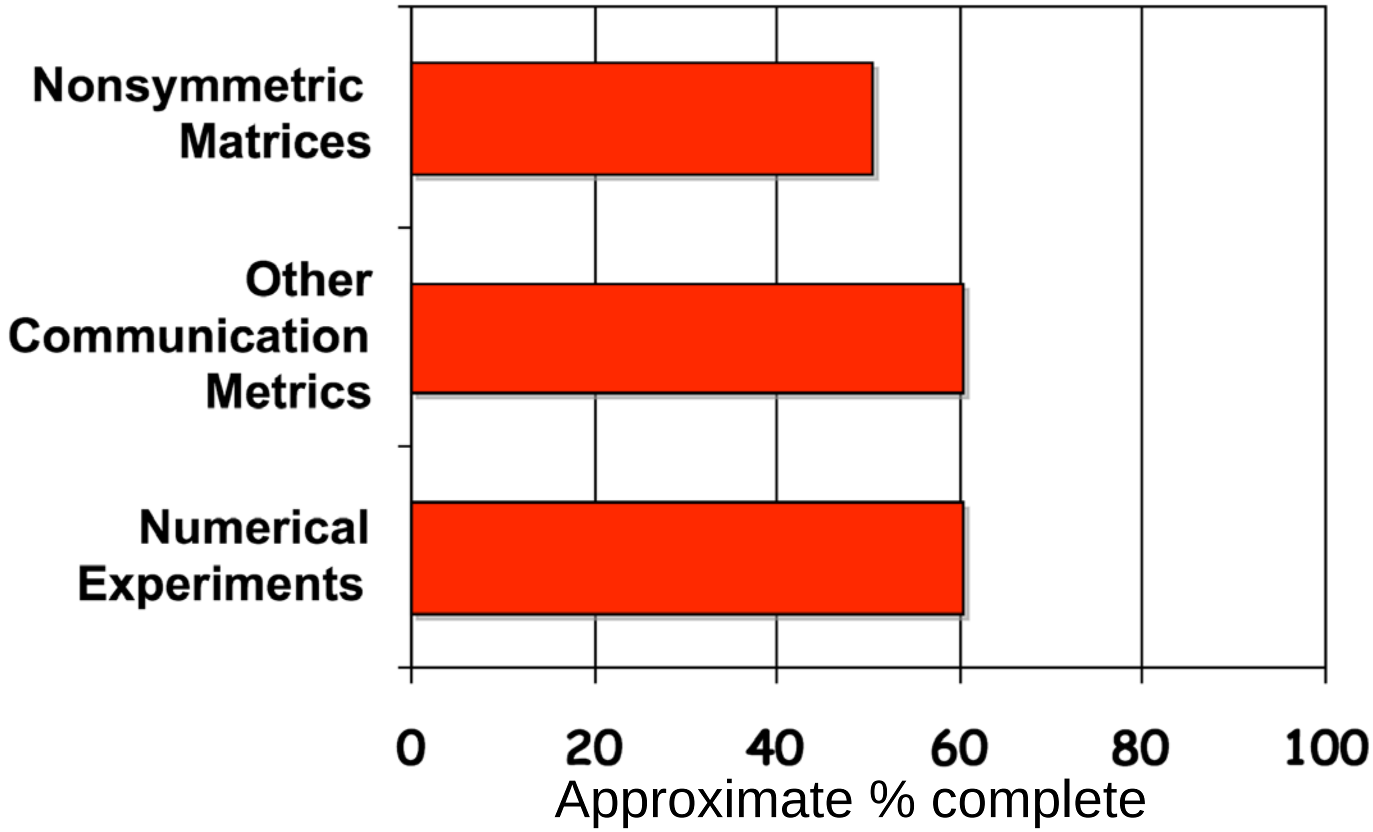
- Corner reordering
  - Implement proposed method
- Nested dissection approach
  - Improve partitioning of separator vertices/edges

Progress: Sparse Matrix Partitioning (2)



- Nonsymmetric matrices
  - Corner method
  - Improve nested dissection approach to nonsymmetric

Progress: Sparse Matrix Partitioning (2)



- Other communication metrics
  - Messages
- Numerical experiments
  - Larger matrices

Acknowledgements/Thanks

- Professor Michael Heath,
  - Advisor
- Dr. Erik Boman, Sandia National Laboratories
  - Summer technical advisor
  - Collaborator on partitioning work
  - Suggested serial matrix-vector optimization problem, hypergraphs, etc.
- Dr. Bruce Hendrickson, Sandia
  - Corner method ordering
  - Suggested vertex-ordering for serial opt. problem
- Professor Robert Kirby, Texas Tech University
  - Serial matrix-vector optimization/FErari discussions

Acknowledgements/Thanks

- Professor Luke Olson
  - Help with FE code to generate matrices
- Professor Jeff Erickson
  - Discussion about serial optimization problem
    - Suggested vertex-ordering
    - Hyperedge detection
- Professor William Gropp
  - Discussion about serial optimization problem
- Funding sources
  - DOE CSGF (Krell Institute)
  - CSCAPES
  - Professor Heath's Fulton Watson Copp Chair